LICENTIATE THESIS

Pattern Recognition with Vector Symbolic Architectures

Denis Kleyko

Dependable Communication and Computation Systems

LULEÅ UNIVERSITY OF TECHNOLOGY

Pattern Recognition with Vector Symbolic Architectures

Denis Kleyko

Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
Luleå, Sweden

Supervisors:

Evgeny Osipov, Wolfgang Birk, Valeriy Vyatkin

Printed by Luleå University of Technology, Graphic Production 2016

ISSN 1402-1757
ISBN 978-91-7583-535-8 (print)
ISBN 978-91-7583-536-5 (pdf)

Luleå 2016

www.ltu.se

To my Family

Abstract

Pattern recognition is an area constantly enlarging its theoretical and practical horizons. Applications of pattern recognition and machine learning can be found in many areas of the present day world including health-care, robotics, manufacturing, economics, automation, transportation, etc. Despite some success in many domains pattern recognition algorithms are still far from being close to their biological vis-a-vis – human brain. New possibilities in the area of pattern recognition may be achieved by application of biologically inspired approaches.

This thesis presents the usage of a bio-inspired method of representing concepts and their meaning – Vector Symbolic Architectures – in the context of pattern recognition with possible applications in intelligent transportation systems, automation systems, and language processing. Vector Symbolic Architectures is an approach for encoding and manipulating distributed representations of information. They have previously been used mainly in the area of cognitive computing for representing and reasoning upon semantically bound information.

First, it is shown that Vector Symbolic Architectures are capable of pattern classification of temporal patterns. With this approach, it is possible to represent, learn and subsequently classify vehicles using measurements from vibration sensors.

Next, an architecture called Holographic Graph Neuron for one-shot learning of patterns of generic sensor stimuli is proposed. The architecture is based on implementing the Hierarchical Graph Neuron approach using Vector Symbolic Architectures. Holographic Graph Neuron shows the previously reported performance characteristics of Hierarchical Graph Neuron while maintaining the simplicity of its design.

The Holographic Graph Neuron architecture is applied in two domains: fault detection and longest common substrings search. In the area of fault detection the architecture showed superior performance compared to classical methods of artificial intelligence while featuring zero configuration and simple operations. The application of the architecture for longest common substrings search showed its ability to robustly solve the task given that the length of a common substring is longer than 4% of the longest pattern. Furthermore, the required number of operations on binary vectors is equal to the suffix trees approach, which is the fastest traditional algorithm for this problem. In summary, the work presented in this thesis extends understanding of the performance proprieties of distributed representations and opens the way for new applications.

Contents

Acknowledgments

This thesis is the result of two years of work at Luleå University of Technology at the Department of Computer Science, Electrical and Space Engineering within the Dependable Communication and Computation Systems group.

First, this thesis would not be possible without the efforts of many people. I would like to express my sincere gratitude to my supervisors: Professor Evgeny Osipov, Professor Wolfgang Birk, and Professor Valeriy Vyatkin. Their belief in me, their passion in scientific discussions, and constant support have helped me to carry out the work presented here. I especially thank Evgeny for being always available to discuss crazy ideas, for his perpetual energy and inspiration, and enormous guidance through the bumpy academic road.

I thank all of the co-authors of the papers included in this thesis: Asad Khan, Arash Mousavi, Nikolaos Papakonstantinou, Ahmet Sekercioglu, and Alexander Senior for their encouragement, valuable suggestions and comments during various stages of research. I also would like to thank Alexander Senior, Siddharth Dadhich, and Sandeep Patil for their valuable comments, which helped to improve the quality of the thesis. Furthermore, I express my gratitude to Blerim Emruli, Ross Gayler, and Fredrik Sandin for fruitful discussions on VSAs.

In addition, I gratefully acknowledge the Swedish Governmental Agency for Innovation Systems (VINNOVA, project 2013-00265), the Swedish Foundation for International Cooperation in Research and Higher Education (STINT, project IG2011-2025), the ARTEMIS Arrowhead project, and the Wallenberg Foundation for the financial support.

Finally, and most importantly, I want to thank my father Vyacheslav and my wife Viktoriia for being a source of constant support, love, and encouragement for me. My thoughts and gratitude are with you.

November 2015, Luleå
Denis Kleyko

Part I

Chapter 1

Thesis Introduction

"All models are wrong, but some are useful."

George E. P. Box

On a high level of abstraction there are two dominant approaches in the area of Artificial Intelligence (AI): symbolic and connectionist. The earlier symbolic approach represents information via symbols and their relations. Symbolic AI solves problems or infers new knowledge through the processing of these symbols. In the alternative connectionist approach information is represented in a distributed, less explicit form within a network of simple computational units. These networks are usually designed to implement parallel constraint satisfaction where the processing is an interaction of diverse knowledge sources.

Nowadays connectionist systems in the area of Artificial Intelligence are perceived as a classical approach. Recently they have been reincarnated after the introduction of deep learning. During the first resurgence of connectionist models in the 1980's particular attention was paid to the way information is presented to a network of connected units. Information representation can be classified into several categories e.g., as in [1]: strictly local; distributed; local; microfeatures; and coarse coding. Strictly local representation uses a dedicated unit to represent a single concept (features, objects, etc.); several active units marks simultaneous presence of several concepts. Local representations have a fixed pool of units, but only one unit can be active at given time representing particular concept. Microfeatures are used when concepts can be decomposed to simpler components (microfeatures), then each concept is represented as an activation of several units where a single unit represents one microfeature in the strictly local manner. Distributed representation encodes a specific concept by a unique pattern of activity over a group of units. Coarse coding is a specific type of distributed representation where units have overlapping activation areas (e.g. in space). A single concept (e.g. a point in space) activates a unit only if it is located in the activation area of this unit.

Despite advantages of localist (local) representations such as explicit representation and simple design of representational scheme they also possess several notable disadvantages [2, 3] as mentioned below:

- Inefficiency of localist representation for large sets of units;
- Proliferation of units in networks that represent complex structure;
- Inefficient and highly redundant use of connections;
- Uncertainty over its capacity to learn elaborate representations for complex structure.

The usage of distributed representations may eliminate these disadvantages. This thesis considers the usage of a specific family of distributed representations in the context of pattern recognition and presents several application areas. Hinton et. al in [4] defines distributed representation in the context of a connectionist system as a representation in which "each entity is represented by a pattern of activity distributed over many computing elements, and each computing element is involved in representing many different entities". In later work [5] Hinton proposes a concept of reduced description. Next, these ideas and other works on convolution memories led to distributed representations formed via tensor products [6]. This became the foundation for an influencing class of distributed representations called Holographic Reduced Representation (HRR) [2, 7, 3].

HRR was the first approach to distributed representations in which the size of the representational dimension was kept fixed. With time the line of investigations related to formation of vector based distributed representations created a research domain of its own and, henceforth, several other approaches have emerged. The commonly accepted name for these approaches is Vector Symbolic Architectures (VSAs). The term was first introduced by Gayler in [8]. One of the initial goals of VSAs [9] is to address the challenges to connectionism posed by Fodor and Pylyshyn [10], Jackendoff [11], and other researchers who demanded to see how, with the aid of connectionist approaches, one could model such crucial features of cognition as compositionality and systematicity.

Practically there are several successful applications using VSAs. Particular examples are Random Indexing [12, 13], which is the computationally lighter alternative to Latent Semantic Analysis and Spaun, which itself is a large-scale model of the functioning brain [14, 15]. Chapter 3 covers some applications of VSAs in more details.

1.1 Problem formulation

Pattern recognition is an area constantly enlarging its theoretical and practical horizons. Applications of pattern recognition and machine learning can be found in many areas of our life including health-care, robotics, manufacturing, economics, automation, transportation, etc. Despite relative success in many domains pattern recognition algorithms are still far from being close to their biological vis-a-vis – human brain. Thus biologically inspired approaches to pattern recognition may open new possibilities in the area.

This thesis presents usage of a bio-inspired method of representing concepts and their meaning – Vector Symbolic Architectures – in the context of pattern recognition with showcases in intelligent transportation systems, automation systems, and language processing. VSAs is an approach for encoding and operations on distributed representation

of information. Among key properties of VSAs are estimation of gradual similarity, efficient use of representational resources, their suitability for usage in associative memories allowing a natural generalization, graceful degradation of representation, resistance to noise and uncertainty, learning from examples, and analogical reasoning. Moreover binary distributed VSAs require low computational complexity and can be potentially deployed on resource constrained devices. The main goal of this work is to investigate the applicability of VSAs for pattern recognition and explore its potential application areas. The specific research questions investigated in this work and addressed in the form of appended papers are:
Q1 Can a VSAs be applied for representation, learning and subsequent classification of temporal patterns in the showcase of a classification of vibration sensors measurements from vehicles?
Q2 Can a VSA-based architecture be capable of forming and memorizing distributed representations for patterns of generic sensor stimuli be created?
Q3 Can the computational task of longest common search benefit from the usage of distributed representations?
Q4 Can such an architecture be successfully applied for fault detection in generic complex systems of systems?

1.2 Thesis outline

This work is a compilation thesis that consists of two parts: Part I is an extended overview of the results (kappa) and Part II includes the appended papers. Part I is organized as follows. Chapter 2 provides the reader with the details of a particular class of VSAs used in this thesis called the Binary Spatter Code. Several application domains of VSAs are discussed in Chapter 3. Chapter 4 is a summary of the appended papers included in Part II. Finally, Chapter 5 concludes the work presented in this thesis and discusses the future work. Part II of this thesis includes four papers: two peer-reviewed papers published in the proceedings of IEEE conferences, one journal paper published in Procedia Computer Science, and one manuscript submitted to IEEE Transactions on Neural Networks and Learning Systems. All papers have been reformatted to match this thesis' layout.

1.3 Summary of appended papers

This thesis includes two conference papers, one journal paper, and one journal manuscript. The papers are presented in Part II. This section briefly summarizes the appended papers and presents authors' contributions to each paper. The last part of the section presents the list of publications which are not included in the thesis.

1.3.1 Paper A

Title: Brain-like classifier of temporal patterns
Authors: Denis Kleyko and Evgeny Osipov
Published in: IEEE International Conference on Computer and Information Sciences (ICCOINS), 2014, Kuala Lumpur, Malaysia
Summary: In this article we present a pattern classification system which uses Vector Symbolic Architectures for representation, learning and subsequent classification of patterns, as a showcase we have used classification of vibration sensors measurements to vehicles types. On the quantitative side the proposed classifier requires only 1 kB of memory to classify an incoming signal against of several hundred of training samples. The classification operation into N types requires only $2 * N + 1$ arithmetic operations this makes the proposed classifier feasible for implementation on a low-end sensor nodes. The main contribution of this article is the proposed methodology for representing temporal patterns with distributed representation and VSA-based classifier.
Author contribution: D.K. and E.O. came up with the initial idea. D.K. implemented the classifier and performed the simulations. The first draft of the manuscript was written by D.K. The final version was written by D.K. and E.O.

1.3.2 Paper B

Title: Holographic Graph Neuron: A Bio-Inspired Architecture for Pattern Processing
Authors: Denis Kleyko, Evgeny Osipov, Alexander Senior, Asad I. Khan, and Y. Ahmet Şekercioğlu
Published in: submitted to Neural Networks and Learning Systems, IEEE Transactions
Summary: In this article, we propose a new approach for implementing Hierarchical Graph Neuron, an architecture for memorizing patterns of generic sensor stimuli, through the use of Vector Symbolic Architectures. The adoption of a distributed representation ensures a single-layer design, while retaining the existing performance characteristics of Hierarchical Graph Neuron. This approach significantly improves the noise resistance of the Hierarchical Graph Neuron architecture, and enables a linear (with respect to the number of stored entries) time search for an arbitrary sub-pattern.
Author contribution: E.O. and A.K. produced the idea of the architecture and wrote the first draft. D.K. proposed improvements to the initial model, implemented it, evaluated the comparison of the proposed architecture with the original method of A.K., conducted the mathematical analysis of the properties of the architecture, and wrote the related sections. All authors participated in the preparation of the final draft and in the process of addressing reviewers' comments.

1.3.3 Paper C

Title: On bidirectional transitions between localist and distributed representations: The case of common substrings search using Vector Symbolic Architecture
Authors: Denis Kleyko and Evgeny Osipov

Published in: Procedia Computer Science
Summary: The contribution of this article is twofold. First, it presents an encoding approach for seamless bidirectional transitions between localist and distributed representation domains. Second, the approach is demonstrated on the example of using Vector Symbolic Architectures for solving a problem of finding common substrings. The proposed algorithm uses elementary operations on long binary vectors. For the case of two patterns with respective lengths $L1$ and $L2$ it requires $\Theta(L1 + L2 - 1)$ operations on binary vectors, which is equal to the suffix trees approach – the fastest algorithm for this problem. The simulation results show that in order to be robustly detected by the proposed approach the length of a common substring should be more than 4% of the longest pattern.
Author contribution: D.K. and E.O. proposed the initial idea during one of scientific discussions. D.K. implemented the algorithm and evaluated its performance. The first draft of the manuscript was written by D.K. The final version was written by D.K. and E.O.

1.3.4 Paper D

Title: Fault Detection in the Hyperspace: Towards Intelligent Automation Systems
Authors: Denis Kleyko, Evgeny Osipov, Nikolaos Papakonstantinou, Valeriy Vyatkin, and Arash Mousavi
Published in: IEEE International Conference on Industrial Informatics (INDIN), 2015, Cambridge, UK
Summary: This article presents a methodology for intelligent, biologically inspired fault detection system for generic complex systems of systems. The proposed methodology utilizes the concepts of associative memory and vector symbolic architectures, commonly used for modeling cognitive abilities of human brain. Compared to classical methods of artificial intelligence used in the context of fault detection the proposed methodology shows a surpassing performance, while featuring zero configuration and simple operations.
Author contribution: E.O., N.P., and D.K. came up with the initial idea of applying VSAs to fault detection. The measurements were taken and processed by N.P. D.K. implemented and evaluated VSAs based approach using the measurements. N.P. benchmarked the traditional algorithms with the measurements. The first draft was written by E.O. All authors participated in the preparation of the final draft.

1.3.5 List of Publications Not Included in the Thesis

1. D. Kleyko, R. Hostettler, E. Osipov, and W. Birk, *Comparison of Machine Learning Techniques for Vehicle Classification using Road Side Sensors*, In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC 2015), pp. 1–6.

2. D. Kleyko, E. Osipov, S. Patil, V. Vyatkin, and Z. Pang, *On Methodology of Im-*

plementing Distributed Function Block Applications using TinyOS WSN nodes, In Proceedings of the 2014 IEEE 19th International Conference on Emerging Technologies and Factory Automation (ETFA 2014), pp. 1–7.

3. D. Kleyko, N. Lyamin, and E. Osipov, *Modified algorithm of dynamic frequency hopping (DFH) in the IEEE 802.22 standard*, ser. Lecture Notes in Computer Science, vol. 8715. Springer, 2014, pp. 75–83.

4. S. Balasubramaniam, N. Lyamin, D. Kleyko, M. Skurnik, A. Vinel, and Y. Koucheryavy, *Exploiting bacterial properties for multi–hop nanonetworks*, Communications Magazine, I E E E, vol. 52, no. 7, pp. 184–191, 2014.

5. O. Melentyev and D. Kleyko, *Computing the parameters of the discrete channel resulting under frequency hopping in the general case*, Automation and Remote Control, Springer, vol. 74, no. 7, pp. 1128–1131, 2013.

6. D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, *Dependable MAC layer architecture based on holographic data representation using hyper-dimensional binary spatter codes*, ser. Lecture Notes in Computer Science, vol. 7642. Springer, 2012, pp. 134–145.

7. E. Osipov, L. Riliskis, D. Kleyko, and N. Lyamin, *Packet-less medium access approach for dependable wireless event passing in highly noisy environments*, ser. Technical report / Luleå University of Technology. Luleå Tekniska Universitet, 2012.

Chapter 2

Binary Spatter Codes

"The theme is that we must rethink computing."

Pentti Kanerva

This chapter presents the theoretical background of a specific class of VSAs called Binary Spatter Codes (BSCs). Binary Spatter codes are used as the basis of the work in this thesis and they have been extensively employed in the appended papers in Part II.

2.1 Early work on BSCs

Spatter Codes were first introduced by Kanerva in [16]. Spatter codes represent a concept (an attribute, a feature, an element of a set, a set, etc.) as two high-dimensional binary random vectors called codewords. One of the two high-dimensional vectors is dense while other being a sparse vector. The dimensionality of vectors N is more than 1000 elements; often $N = 10000$ is used. The dense representation of the concept may be used in an associative memory. The sparse vectors of low-level components are used to create high-level concepts by being summed together forming the superposition. The sparse vector for the new high-level concept is the superposition thresholded at 1.5.

The dense vector is created using logical OR operation on low-level components' sparse codewords. The approach is partially inspired by the psychological theory of chunks, where a chunk can be constructed from other chunks. Spatter codes represent chunks as large patterns of bits. When constructing new chunks there are two requirements [16]:

- as several chunks combine to form a new chunk, the codewords for the components should be visible in the codeword for the new chunk (satisfied by the dense vector);
- the new chunk should be represented by a codeword that can be used in the construction of codewords for further chunks (satisfied by the sparse vector).

Thus spatter codes provide a simple way for constructing new chunks as a combination of old ones. However, it is unconventional to have two vectors representing only one concept (chunk).

Starting from [17], Kanerva used only one vector for the representation of an item where he studied the properties of BSCs. In that paper, the statistical properties of spatter codes were derived from the binomial distribution assuming that bits are independently distributed. The key property that this allows is recursiveness; in this context, recursiveness means that the probability of ones (p) in codewords of components is the same as in the compound vector. Different levels of thresholds for the superposition of codewords with probability $p = 0.5$ were applied. For $p = 0.5$ if K components are composed the threshold $K/2$ yields recursive spatter code. Hamming distance normalized to the length of vector (δ) is used to measure how close two vectors are to each other. The distance between two independent codewords is $2p(1 - p)$. However, when compound vectors have components in common their distance is smaller. Table 3 in [17] shows how distances change with the number of components in common.

The last part of the paper deals with visibility of components. There is no easy way to extract the components of a compound vector; however, given two codewords one can measure their distance and see whether one of them is a component of the other. At the same time it is impossible to recognize which is the component of which and thus additional mechanisms are needed, for example clean-up memory, which will be discussed in the next section.

2.2 BSCs as a binary analogy to HRR

The first works on spatter codes [16, 17] only considered representation of sets. A binary analogy to HRR was introduced in [18]. This paper presented the method of encoding and decoding structured information with random binary vectors. As an example of structured information, they considered a medical record consisting of name, gender, age, and weight (Joe, male, 66, 77). Names of fields in the record were roles (variables) and particular values were fillers. In order to represent a record in binary spatter codes, both roles and fillers were assigned random binary vectors ($p = 0.5$). In other words, codewords act like pointers in traditional computing architecture, i.e. they can represent any entities [18]: objects, concepts, features, sets of features, attributes, values, role-filler pairs, data structures, etc. Role-filler pairs are composed and decomposed via the bitwise Boolean Exclusive-OR (XOR, $\otimes$) operation. Set of role-filler pairs are encoded via applying a threshold to the superposition of components. The operation is denoted as $[\mathbf{a} + \mathbf{b} + \cdots + \mathbf{c}]$, where the brackets $[\cdots]$ represent the thresholding, vectors are indicated in bold: $\mathbf{a}$, $\mathbf{Joe}$, $\mathbf{age}$. Note that a spatter code with a threshold of $K/2$ is recursive. As a consequence of this the number of components K in the spatter-coded sets should be an odd number, otherwise ties are broken at random (which corresponds to adding a random vector to the set). Thus, the record for JSmith will be represented as: $\mathbf{JSmith} = [\mathbf{name} \otimes \mathbf{Joe} + \mathbf{sex} \otimes \mathbf{male} + \mathbf{age} \otimes \mathbf{66} + \mathbf{weight} \otimes \mathbf{77}]$. The ordered record can be represented if codewords for roles corresponds to positions in the record. Extraction of fillers from the record is done via XOR of the record's codeword and role's codeword, e.g. to know gender of a patient: $\mathbf{sex} \otimes \mathbf{JSmith}$. The result is approximately male, because other components like $\mathbf{sex} \otimes \mathbf{age} \otimes \mathbf{66}$ are not valid codewords, i.e. they

Table 2.1: Records for holistic encoding

Record	name	sex	age
PF3	Pat	female	33
PM6	Pat	male	66
LF6	Lee	female	66
LM3	Lee	male	33

act as a random noise. Thus, the decoding process is noisy (vectors are approximate) and requires an additional mechanism in order to extract the original codeword. This mechanism is called clean-up memory: it is an autoassociative memory that stores all currently defined (valid) codewords as fixed-point attractors. Thus, the clean-up memory takes noisy codewords as inputs (e.g. **sex** $\otimes$ **JSmith**) and retrieves the best-matching noise-free codewords (e.g. **male**).

2.3 Holistic mapping

In [19] Kanerva presents a tutorial on binary spatter codes rich with examples on forming distributed records as an alternative to the standard localist representation. Distributed representations do not have fields as such and therefore they are also called holistic, which means that each bit in the vector contains some information about every field of a record. From the entire family of binary spatter codes Kanerva [19] choses the equally probable distribution when bits in the codeword are independent, and $p = Pr\{1\} = Pr\{0\} = 0.5$. Formation of a holistic record is a two-step process. The first step is the binding that combines roles and corresponding fillers; an important property of the operation is that the binding of two random vectors produces a random vector that resembles neither of the two. At the second step role-filler bindings are composed into single vector; this operation has several names: chunking, merging, superimposing, bundling. Recursive binary spatter codes with $p = 0.5$ use the majority rule for the bundling operation. Components of the bundling operation are similar to the compound vector. The similarity between two vectors is characterized by normalized Hamming distance, which (for two vectors) measures the number of positions in which they differ. If all components are random and independent then the expected normalized Hamming distance between a component $\mathbf{X}$ and the compound $\mathbf{A}$ is $\delta(\mathbf{X}, \mathbf{A}) = \frac{1}{2} - \frac{\binom{K-1}{(K-1)/2}}{2^K}$, where K is assumed to be an odd number. Naturally the more components there are in a holistic record the less similar a single component is to the compound vector. The standard deviation (σ) of the expected distance depends on the dimensionality and is calculated as $\sigma = \sqrt{(\delta(1-\delta))/N}$. The distance δ is related to the correlation coefficient (normalized covariance) by $\rho = 1-2\delta$. The extraction of a part from the holistic record is called probing. In most cases it involves unbinding, which is equivalent to binding for binary spatter codes, and querying the clean-up memory. The paper presented several interesting examples of processing with holistic representation. Records used in [19] as examples are shown in Table 2.1.

An example query to the system by the user can be "What is the age of Lee who is a female?". The correct record for this query will be **LF6** and the answer is 66. Kanerva considered three cases for such an example query which are as follows:

Case 1. This case assumed that only the codewords **name**, **Lee**, **sex**, **female**, and **age** are known. *Solution* 1. **LF6** is approximated by replacing the unknown $\mathbf{age} \otimes \mathbf{66}$ component with a random vector. The clean-up memory will return **LF6** as the closest solution. Next it can be probed with **age**; the memory issues **66** as the result.

Case 2. In this case only **Pat**, **male**, **PM6**, **Lee**, **female**, and **age** are known. *Solution* 2a. First find **name** and **sex** via probing **PM6** with **Pat** and **male**, then apply the previous solution. *Solution* 2b. This solution uses the correspondences $\mathbf{Pat} \leftrightarrow \mathbf{Lee}$ and $\mathbf{male} \leftrightarrow \mathbf{female}$ by forming the mapping $\mathbf{T} = [\mathbf{Pat} \otimes \mathbf{Lee} + \mathbf{male} \otimes \mathbf{female}]$. Interestingly probing the vector **T** with **PM6** returns **LF6** (see [19] for a detailed explanation). *Case* 3. Only **33**, **PF3**, and **LF6** are known. *Solution* 3. One step solution is to probe **LF6** with $\mathbf{33} \otimes \mathbf{PF3}$. The answer will be ambiguous, but it exemplifies possibility of holistic processing without intermediate steps. Thus, *Solutions* 2b and 3 are examples of a different type of computing, called, holistic mapping, i.e. mapping between points in a high-dimensional space. There is no equivalent to this kind of processing using geometric properties of the representational space in the traditional computing. However, holistic mapping can be seen as a parallel alternative to traditional sequential search.

Other examples of holistic mapping are given in [20, 21]. In [20] two records are used as the examples. The first record **F** is a distributed representation of France: the capital (**ca**) is Paris (**Pa**), the geographic location (**ge**) is Western Europe (**WE**), and the monetary unit (**mo**) is Euro (**er**). The holistic pattern for France is created as: $\mathbf{F} = [\mathbf{ca} \otimes \mathbf{Pa} + \mathbf{ge} \otimes \mathbf{WE} + \mathbf{mo} \otimes \mathbf{er}]$. The second record **S** is a distributed representation of Sweden: the capital (**ca**) is Stockholm (**St**), the geographic location (**ge**) is Scandinavia (**Sc**), and the monetary unit (**mo**) is krona (**kr**). The holistic pattern for Sweden is $\mathbf{S} = [\mathbf{ca} \otimes \mathbf{St} + \mathbf{ge} \otimes \mathbf{Sc} + \mathbf{mo} \otimes \mathbf{kr}]$. Holistic mapping allows us to find the answer to the question "What is the Paris of Sweden?". First create "Paris of France" $\mathbf{F} \otimes \mathbf{Pa}$ (approximately **ca**), then probing with $\mathbf{S} \otimes \mathbf{F} \otimes \mathbf{Pa}$ will return Stockholm as the best match. Thus, the example shows that holistic mapping is capable of answering analogy questions. Furthermore, it also allows multiple substitutions at once for example, the mapping vector for Sweden and France is created by binding the corresponding items to each other and forming a holistic record as: $\mathbf{M} = [\mathbf{Pa} \otimes \mathbf{St} + \mathbf{WE} \otimes \mathbf{Sc} + \mathbf{er} \otimes \mathbf{kr}]$. Thus, the binding of the record **F** with **M** creates a noisy version of the holistic record of Sweden. An example of construction of the noisy version of the mapping vector for two holistic vectors of USA ($\mathbf{USSTATES} = [\mathbf{nam} \otimes \mathbf{USA} + \mathbf{cap} \otimes \mathbf{WDC} + \mathbf{mo} \otimes \mathbf{DOL}]$) and Mexico ($\mathbf{MEXICO} = [\mathbf{nam} \otimes \mathbf{MEX} + \mathbf{cap} \otimes \mathbf{MXC} + \mathbf{mo} \otimes \mathbf{PES}]$) is presented by Kanerva in [22]. In this case the mapping vector for USA and Mexico is created as $\mathbf{M2} = \mathbf{USSTATES} \otimes \mathbf{MEXICO} = [\mathbf{USA} \otimes \mathbf{MEX} + \mathbf{WDC} \otimes \mathbf{MXC} + \mathbf{DOL} \otimes \mathbf{PES} + \mathbf{noise}]$. Then the answer to the question "What is the dollar of Mexico?" can be found by $\mathbf{DOL} \otimes \mathbf{M2}$ because the result of the operations is a noisy version of **PES**. Interestingly, if there is another mapping vector, e.g. $\mathbf{M3} = \mathbf{USSTATES} \otimes \mathbf{SWEDEN}$ then the mapping between **M2** and **M3** cancels out **USSTATES**. The result can be seen as

interpretation of Sweden in terms of Mexico or vice versa. Kanerva pointed out that this mapping resembles the problem of translation a text from one language to another through a third one.

2.4 Pattern Completion with BSCs

The ability of distributed representations created with binary spatter codes to complete generic patterns is studied in [23]. In this paper Kanerva showed that despite the absence of fields in holistic records their performance is similar to connectionist models in a conventional pattern-completion task. A drugdata base was used for demonstration. An advantage of distributed representation is that there is no need to take care of the alignment of fields. In the conclusion, Kanerva suggested that sparse distributed representations can also be of interest in this field.

2.5 Learning from example

The capability of HD-codes to learn mappings between concepts from examples was presented in [24, 21]. This follows from the self-inverse binding property of Binary Spatter Codes (and some other VSAs). $\mathbf{A} \otimes \mathbf{B}$ represents the binding of $\mathbf{A}$ with $\mathbf{B}$, but it can also be interpreted as a representation of a mapping from $\mathbf{A}$ to $\mathbf{B}$ and vice versa. Binding $\mathbf{A} \otimes \mathbf{B}$ with $\mathbf{A}$ yields $\mathbf{B}$ and vice versa, hence $\mathbf{A} \otimes \mathbf{B}$ can be interpreted as the representation of a mapping that when applied to $\mathbf{A}$ transforms it to $\mathbf{B}$ and vice versa. Learning by example can be seen as a form of relational learning. It is based on the property of BSCs to keep structural similarity of the represented knowledge structures. In essence the key component is a mapping vector $\mathbf{M}$ which acts as a mapping structure to infer similarity between compositional structures. We use an example from [24] to demonstrate this concept here. It is known that if a is the mother of b, then a is the parent of b. Both data structures are represented using holistic records. For the mother relation, we have the structure $\mathbf{mother}_{AB} = [\mathbf{mother}+\mathbf{mot}\otimes\mathbf{A}+\mathbf{child}\otimes\mathbf{B}]$, where **mother** is the HD-code representing the relation for mother, **mot** and **child** are HD-codes for roles in this structure. For the parent relation $\mathbf{parent}_{AB} = [\mathbf{parent}+\mathbf{par}\otimes\mathbf{A}+\mathbf{child}\otimes\mathbf{B}]$, where **parent** is the HD-code representing the relation for parent, **par** and **child** are HD-codes for roles in this structure. The vector $\mathbf{M}$ in turn is constructed as binding of $\mathbf{parent}_{AB}$ and $\mathbf{mother}_{AB}$, i.e. $\mathbf{M} = \mathbf{M}_{AB} = \mathbf{mother}_{AB} \otimes \mathbf{parent}_{AB}$. Thus, the mapping vector $\mathbf{M}$ is constructed from the particular example of mother-parent relation. Furthermore, the mapping vector $\mathbf{M}$ can be updated for new examples as: $\mathbf{M} = [\mathbf{M}_{AB} + \mathbf{M}_{CD} + \mathbf{M}_{EF} \cdots]$, where c, d, e, f, are new examples of the same relations. It is possible to generalize that x is the parent of y given only that "x is the mother of y" in the form $\mathbf{mother}_{XY}$ by unbinding $\mathbf{M} \otimes \mathbf{mother}_{XY}$ for previously unseen fillers $\mathbf{X}$ and $\mathbf{Y}$. The result of unbinding will be most similar to $\mathbf{parent}_{XY}$. This behavior is due to the fact that the same literal $\mathbf{A}$ has been in both relationships, $\mathbf{mot} \otimes \mathbf{A}$ and $\mathbf{par} \otimes \mathbf{A}$. The binding $\mathbf{M}_{AB}$ contains terms like $(\mathbf{mot} \otimes \mathbf{A}) \otimes (\mathbf{par} \otimes \mathbf{A}) = \mathbf{mot} \otimes \mathbf{par}$, which implement the mapping of the roles

independent of the new fillers. Simulation results in [24] showed that good generalization is already achieved for three different examples in the mapping vector. However, the disadvantage of such mapping is its bidirectionality which is due to the fact that the unbinding operation is equivalent to the binding operation. For example the mapping of parent-of to mother-of is valid, but such an inference is wrong.

2.6 Encoding of sequences

The encoding of sequences into the distributed representation can be done by random permutations as presented in [25, 26]. A permutation is similar to binding because it also distributes over bundling; it distributes over any bitwise operation including binding; it preserves distance; and it is invertible. In contrast to the binding in BSCs permutation is not its own inverse, but every permutation has an inverse matrix. Several repetitions of the same permutation create a new orthogonal vector, and therefore it is suitable for sequence encoding. One possible scheme is to superimpose a permuted version of its history to each element of the sequence , and then store the superimposed vector. For example the encoding of the sequence of vectors ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\cdots$) with Π denoting a permutation proceeds as follows:
$\mathbf{S1} = \mathbf{A}$
$\mathbf{S2} = \Pi\mathbf{S1} + \mathbf{B} = \Pi\mathbf{A} + \mathbf{B}$
$\mathbf{S3} = \Pi\mathbf{S2} + \mathbf{C} = \Pi(\Pi\mathbf{A} + \mathbf{B}) + \mathbf{C}$
$\mathbf{S4} = \Pi\mathbf{S3} + \mathbf{D} = \Pi(\Pi(\Pi\mathbf{A} + \mathbf{B}) + \mathbf{C}) + \mathbf{D}$.
Taking into account distribution over the superposition, **S4** will simplify to:
$\mathbf{S4} = \Pi\Pi\Pi\mathbf{A} + \Pi\Pi\mathbf{B} + \Pi\mathbf{C} + \mathbf{D} = \Pi^3\mathbf{A} + \Pi^2\mathbf{B} + \Pi\mathbf{C} + \mathbf{D}$.
Thus, the number of permutations of a single vector counts how far it appears in the sequence. With such encoding a sequence can be found in memory only using a short subsequence as an input query. In [26] this approach to sequence encoding is called representing sequences by permuting sums.

For a summary of the work done in VSAs in general and BSCs in particular, interested readers are directed to [26]; there, Kanerva presents motivation for cognitive (hyper)computing, highlights a number of biologically plausible properties of VSAs models, introduces operations on BSCs, showcases applications of VSAs and provides discussion on future development of the paradigm.

Chapter 3

Applications of Vector Symbolic Architectures

To date the development of Vector Symbolic Architectures mostly falls into the research domain. The number of applications using VSAs for solving real-world problems is limited. This chapter first presents one of the most well-known applications of distributed representations with vectors called Random Indexing. Random Indexing is a method for construction of words' semantic models using large learning text corpora.

The second part of this chapter describes the recent advances in using VSAs for modeling of statistical dependencies in temporal sequences as well as for the encoding of continuous values into high-dimensional distributed representations. The methods were exemplified using several application areas including activity recognition using accelerometer data, spoken words recognition, and prediction of the next state (e.g. the next app to be loaded) using real-life mobile phone user data.

These methods and applications are directly related to the future development of this thesis. Papers in Part II of this thesis deals with discrete and static data. The inclusion of techniques allowing to process continuous temporal data will significantly broaden out the range of possible application areas.

3.1 Random Indexing

Random Indexing (RI) was proposed in [12] as an alternative to Latent Semantic Analysis (LSA). LSA is used for computing high-dimensional semantic (context) vectors for words from their co-occurrence in documents. The size of the vocabulary in experiments is around 60000-80000 words. LSA is based on counting occurrence in each word in the corpus of documents. Next the dimensionality of vectors is reduced by Singular-Value Decomposition. The resulting semantic vectors capture meaning; for example they were successfully applied for TOEFL (Test of English as a Foreign Language) synonym test. RI assigns a sparse ternary index vector for each document and a context vector for each word. An index vector consists of 1800 elements with several randomly placed

nonzero elements {-1, +1}. The context vectors of words are of the same dimensionality as index vectors and are initially empty. Every time the word occurs in a document the index vector of the document is added to the context vector. In the synonym test RI showed performance similar to LSA. However, the advantage of RI over LSA is that it avoids usage of Singular-Value Decomposition for the reduction of dimensionality of the semantic vectors; additionally, the dimensionality of a context matrix depends only on the size of a vocabulary. RI can also be applied in the case when a context window is used instead of the whole document. In [27] Random Indexing was studied using TASA corpus and TOEFL synonym test. In the experiments both the dimensionality of vectors ($N = 1000 - 100000$) and number of nonzero elements ($M = 2 - 60$) was varied.

Context vectors, an approach similar to Random indexing, was described in [28] by Gallant. The process of Context Vectors generation is different, but also uses random vectors as initial points in a representational space. Both approaches are able to fulfill similar goals: capturing semantic similarity among words; fast querying of constituent objects; automatic generation of representations from an unlabeled corpus; suitability of representations for traditional learning algorithms. Notably Gallant emphasized a need for a Grand Unified Representation which would be able to represent syntax, discourse and logic.

Note that representations created by RI are numeric vectors. In order to use them, for example for Associative-Projective Neural Networks [29] structures, binarization is required. Misuno et al. [30] proposed using thresholds to create binary or ternary representations; the density of vectors can be maintained by adjusting thresholds. Different representational schemes were compared upon their ability to represent semantic similarity of words. Several metrics for similarity calculation were considered: Jaccard similarity, mutual information, Euclidean distance, Manhattan distance, etc. Representations of words were created using TASA, BNC, and Altavista corpora for training. A number of experiments such as: TOEFL synonym test, ESL synonym test, automatic text translation, etc. were performed. One of the conclusions was that there are parameters for context vectors which result in discrete context vectors which perform similarly to numeric context vectors; however, the performance of binary vectors was slightly less than that of ternary vectors.

It was shown in [31] that distributed representations created with RI can be used for efficient automatic text search and classification, for example, the return of most relevant texts for a query. It is achieved through a combination of classification algorithms and distributed representations of words. Experiments in [31] were performed with SVM, kNN and perceptron algorithms. The classification task was demonstrated on Reuters-21578 corpus. The highest accuracy was demonstrated for SVM classifier. For the task of text search biomedical corpus MEDLARS was used. RI accuracy was between the highest results for different vector models.

The problem of automatic text search with distributed representations was further elaborated in [32]. The prototype of the search engine was proposed. Four main steps include: corpus indexing, matrix formation, context vectors formation and search of texts. MEDLARS, Cranfield, TASA, and Time Magazine corpora were used in experiments.

The average accuracy of text search based on context vectors formed by RI can reach 70%. It was confirmed that discrete context vectors demonstrate performance similar to representations with real numbers.

Investigations on text classification with distributed representations were continued in [33, 34]. Experiments for different types of data were performed; in particular numeric vectors, text and images were considered. Numerical vectors were studied in tasks with nonlinear classes, e.g. for Elena dataset. Vectors were encoded using RNC and Prager schemes (see [35] for the detailed description) using a variety of parameters. Distributed representations of vectors served as input data for classification algorithms: SVM with different kernels, perceptron. Distributed representations of words were tested on the Reuters-21578 corpus using SVM, and distributed representations of images were used to classify the MNIST database. Authors concluded that using distributed representations reduced computational expenses while maintaining high classification accuracy.

The original RI method includes the contexts in which focus word appears, but it does not include order of words into their distributed representations. However, the inclusion of order information may represent the grammatical role of a word, and thus reinforce the distributed representation. The possibility of inclusion of word order into semantic vectors using holographic reduced representations by means of circular convolution was shown in [36] by Jones and Mewhort. Inspired by a method in [36] Sahlgren, Holst, and Kanerva in [37] proposed encoding the order of words into RI by forming semantic representations using permutations of vector coordinates. The advantage of the permutation (denoted as Π) based approach is its computational simplicity compared to circular convolution. The permuted vector $\Pi\mathbf{a}$ becomes nearly orthogonal to the original version $\mathbf{a}$. The permutation is invertible, i.e. the original vector can be recovered with the inverse permutation (denoted as Π^{-1}) as $\Pi^{-1}\Pi\mathbf{a}$. The principle of encoding order information in RI with permutations can be exemplified with a simple sentence: "a dog bit the mailman". For example, if the word "dog" is in the focus the order information is encoded as: $\langle\mathbf{dog}\rangle = \Pi^{-1}\mathbf{a} + 0 + \Pi\mathbf{bit} + \Pi^2\mathbf{the} + \Pi^3\mathbf{mailman}$, where Π^n means that the index vector of a word is permuted n times. Thus the order of permutation of the index vector indicates how far it is from the focus word, e.g. in the example "dog" is immediately followed by "bit". RI with permutations encoding allows retrieving frequent neighbors of a word using its semantic vector. In the proposed approach the order can be included in two ways. The first modification includes only words occurring before and after the focus word. This modification is called direction vectors. The second modification called order vectors includes all order information inside the processing window as in the example above. In the experiments RI context vectors, direction vectors, and order vectors were constructed from the TASA corpus. The accuracy of methods was tested for a TOEFL synonym test. The results showed that that direction vectors achieved 80% correct answers on the test. It is a significant improvement in comparison to the performance of RI context vectors. Thus the authors concluded that the permutation of vector coordinates is a viable method for encoding order information in word space.

In [38] the authors compared two approaches of order encoding for semantic vectors, namely random permutations based (denoted as RPM) as in [37] and circular convolution

as in [36] (denoted as BEAGLE). The encoding of word order is an attempt to improve the "bag of words" approach which is often criticized for its disregard of word-order information. Simulations showed that the storage capacity and the probability of the correct decoding are similar for both approaches, but the storage capacity of random permutations drops off slower as the dimensionality is reduced. Another experiment assessed two versions of BEAGLE (originally with circular convolution and then modified with random permutations) on a set of semantic tasks using a Wikipedia corpus for the formation of semantic vectors. Both approaches showed similar performance on a random subset of the corpus. Note, however, that the permutation operation features lesser computational complexity than circular convolution. Therefore only the modified version of BEAGLE was able to process the whole corpus showing the advantage of random permutations to scale to large text corpora. The usage of the whole corpus significantly improved performance. In the last part the BEAGLE method was compared to RI with permutations [37] showing similar performance on the small corpus. The two approaches were further compared in [39] confirming the results in [38]. The new experiments proved the feasibility of using non-unique replacements (units that are mapped to other units arbitrarily) in random permutations. There were no significant differences in performance of the two modifications. This result increased the neurological plausibility of the encoding with random permutations. Thus, based on the results the authors concluded that random permutations are a promising and scalable alternative to circular convolution.

Promising results of application of RI for identification of the language of text samples were presented in [40]. The distributed representation of the language is formed by RI from letters blocks (q-grams) encountered in a text sample. The prototypical semantic vectors of languages are precomputed in advance using the same method but for larger corpora. Then the representation constructed from a current text sample is compared with all prototypical language vectors. The prototypical vector with the highest similarity score indicates the most likely language for the current sample. The method was tested for 21 different languages using Project Gutenberg Hart and Wortschatz Corpora achieving 97.8% accuracy.

3.2 Modeling of statistical dependencies in sequences

In [41] Räsänen and Kakouros applied the principles of hyperdimensional computing for modeling of statistical dependencies in multivariate discrete sequences. The proposed approach called hyperdimensional computing based predictor (HDCP) is able to treat several discrete sequences in parallel. The main objective of the approach is to predict the future states of streams of sequences given the history of previous states in streams. The HDCP encodes the current and preceding states (up to the specified lag value, m) of all parallel input streams into a single hyperdimensional vector called the context vector. Note that it is assumed that every stream is in time and has a finite alphabet of states. The HDCP represents every state by a random dense vector of +1s and -1s. Random vectors are also assigned to encode the relative temporal positions (from 0 to m) in streams. Thus the binding between the state representation and the representation of position in

the stream forms a unique vector for the given discrete state and temporal delay. The binding is used to encode the temporal structure of the states. The HDCP creates the context vector by bundling bound vectors across time and across modalities. The HDCP, however, uses a weighted bundling operation, where the coefficient for each component is determined by the mutual dependencies between the inputs. For the estimation of these coefficients for the given input streams, a mutual-information (MI) function is used (see Section II in [41]). Thus, the result of the weighted bundling represents the current cross-modal and temporal context in a single context vector. During the training of the HDCP the context vector is used to update a model of the next state in the training data. The prototype model vector is updated by summing up the current vector predicting the state with a newly observed context vector which leads to this state. The learned vectors are used in the prediction stage. During the prediction the HDCP estimates the similarity between the current context vector and vectors in the learned model. The prediction is the state whose vector is the most similar to the current context vector. In other words, calculated similarities estimate the probability distribution of the forthcoming states from the currently observed states across the streams. The HDCP approach was tested in an activity recognition task with data from several body sensors using the Palantir Context Data Library. The demonstrated performance of the approach improves accuracy compare to the previously documented baseline for the task. Also, by using the MI coefficients the HDCP automatically accounts for the varying reliability of different input streams at different temporal delays. In other words the streams' states are utilized only if they have predictive power for the presently predicted state. Moreover with the availability of new data the prediction model could be learned on the fly.

Further work by Räsänen and Saarinen in [42] provides a detailed study of the prediction of the next state for the case of a single discrete stream. The approach proposed in [42] is similar to the one in [41] for the case of a single stream. However, in [42] instead of dense hyperdimensional vectors the authors use sparse vectors with ternary elements {-1; 0; 1}. The sparseness of the approach was inspired by Random Indexing. The density of the non-zero elements was set to 5% of vector's dimensionality. The approach is called sparse distributed predictor (SPD), due to the sparse vectors it uses. When the SDP builds the context vector for the currently observed sequence a weighted bundling is also applied, where the coefficients for different temporal delays are calculated using MI. The process of building the prediction model of each possible state in the stream is the same as in [41], which is also similar to Random Indexing [12]. The prediction of the SDP is the state whose prototype model vector has the largest similarity with the current context vector. Note that prototype model vectors can be normalized in two ways: columnwise and rowwise. The performance of the SDP is shown using predictions from real-life mobile phone user data. The user data includes music playback logs, application launch logs and, sequences of GPS locations. The results of experiments show that SDP outperforms n-grams and mixed-order Markov models in terms of unweighted average recall. The accuracy for weighted average recall is comparable with a mixed-order Markov chain. The training of the SDP, however, is incremental, i.e. it does not require many iterations allowing near real-time online learning and, therefore, can be applied in

resource constrained devices. As concluded by the authors in [42] the major limitation of SDP is that it only learns an approximation of Markov processes up to some finite order using episodic descriptions of the sequences. Therefore, the approach fails in capturing complex long-distance regularities, such as those present in embedded grammars.

In [43] Räsänen proposed an encoding scheme for transforming continuous valued feature vectors into distributed representations. Most of the work in the area of VSAs considered only discrete categorical data. Therefore, the problem of mapping of continuous-valued data into hyperdimensional space is important for applying VSAs in a broader class of real world problems with sensory data. The mapping function from low-dimensional space into hyperdimensional space should satisfy the following requirements [43]: local similarities between input vectors must be approximately maintained, enabling generalization towards new input tokens; distant inputs should be coded with orthogonal vectors; a continuous distance metric between original vectors should be also continuous and smooth in the hyperspace; the local/distant trade-off in the previous requirements should be controllable. The requirement for maintaining relative distance can be accomplished by a linear expansion of the original feature vector onto a binary random projection matrix {-1; +1}. However, one matrix cannot satisfy the requirement on orthogonality of distant vectors. Thus, Räsänen proposed to deterministically encode the original vector by projecting it onto several random matrixes. The distributed representation is a weighted linear combination of these projections. The weight coefficient for every matrix controls the rate of change from one projection to another and it is determined by the original feature vector itself (see equation (4) in [43]). This encoding scheme is called Self-regulated Weighted Accumulation of Random Projections (S-WARP). S-WARP was applied in a spoken word recognition task; the task was to classify a spoken word from a small vocabulary of 79 words using the CAREGIVER Y2 UK corpus as a source of data. The results showed that S-WARP's performance is comparable to a Gaussian mixture-based continuous-density hidden-Markov model.

Chapter 4

Pattern Recognition with Vector Symbolic Architectures

This chapter provides an overview of the results described in details in the papers included in the second part of the thesis.

4.1 Paper A: Brain-like classifier of temporal patterns

Paper A demonstrates the capability of VSAs to represent and classify patterns. By patterns we mean signals from different types of sensors in response to external events featuring repeated patterns in time. For example a vehicle passing generates vibration sequences unique to different types of vehicles; another example on a longer time scale would be measurements of typical temperature changes in a specific environment. The main contribution of this article is the proposed methodology illustrated in Fig. 4.1 for encoding signal patterns using distributed representations and a VSA based classifier. Quantitatively, the proposed classifier requires approximately 1 kB of memory to classify an incoming signal against several hundred training samples. Classifying a signal into one of N types requires only $2 * N + 1$ arithmetic operations, which makes the proposed classifier feasible for implementation on low-end sensor nodes.

The solution presented in this article showcased a real-life scenario using raw measurements of vehicle passages collected from a vibration sensor installed on the road surface for classifying the vehicle's type. For example consider the filtered vibration signal from a 2-axle truck exhibiting a certain pattern in time illustrated in Fig. 2. The task of raw sensory signal conversion into VSA represented pattern is formulated as finding and quantifying distinct changes in signal level and representing them using HD-vectors. For example the signal in Fig. 2 features changes at 9 positions. The processing algorithm produces a *series* of values for magnitudes of *significant* signal's changes. The values of the magnitudes are quantized into a finite number of discrete quantization levels as

illustrated in Fig. 2.

The stages for constructing a VSA based classifier are: 1.) encoding of detected patterns of significant signal changes into VSA representations; and 2.) bundling several VSA-represented patterns into a single VSA representation (HD-vector), which will serve as the classifier. The main goal of the encoding task is to make the VSA representation of patterns which are different even in a single position mutually dissimilar. It is achieved by XOR-ing HD-vectors encoding individual elements when constructing the VSA representation of a particular pattern.

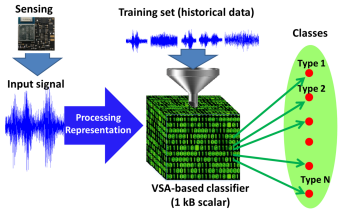


Figure 4.1: System architecture of the VSA-based classifier.

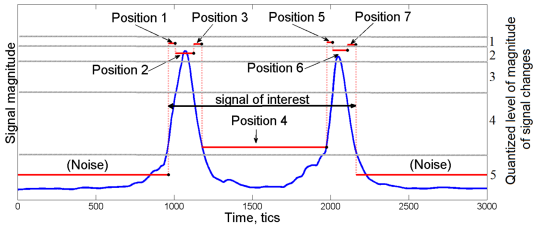


Figure 4.2: An example of signal ready for VSA representation.

We build our classifier in Eq (1) following a similar line of reasoning as in [44]. In (1) the SUM is the MAJORITY operation, P_j is the VSA-represented pattern of a signal, and T_i is an HD vector representing the type to which the signal belongs to. In our example scenario we have three types of signals: a passenger car, a two-axle truck and a three-axle truck.

$$\text{CLASSIFIER} = \text{SUM}_{j=1..N} P_j \otimes T_i | \{i = 1..K\}. \tag{4.1}$$

When the classifier is constructed by bundling all patterns bound to their corresponding classes, it is ready for determining the type of a pattern (P) by testing its inclusion in the VSA classifier. The result of the classification is the class to which the input pattern belongs to: Type $= P \otimes$ CLASSIFIER.

The result of the above operation is the noisy version of the HD-vector encoding the type that the pattern belongs to. The remaining operation is therefore performing a Hamming distance test with the clean versions of the HD-vectors representing types.

Applied to our showcase scenario, the results show that the Hamming distance test for each pattern correctly identifies the type that the pattern belongs to ($\Delta_H << 0.5$). This result confirms the hypothesis for suitability of the VSA for classification of patterns. Thus, it is demonstrated that even without the self-learning capabilities, the proposed classifier has real-life applications. In some applications the proposed classifier can be implemented in low-end hardware while still demonstrating sufficient performance.

4.2 Paper B: Holographic Graph Neuron: A Bio-Inspired Architecture for Pattern Processing

This article presents contributions in two domains: Organization of associative memory, and properties of connectionist distributed representation. In the first area the article introduces a novel bio-inspired architecture, called Holographic Graph Neuron (HoloGN), for one-shot pattern learning, which is built upon Hierarchical Graph Neuron's (HGN) [45] flexible input encoding abstraction and the strong reasoning capabilities of the VSA representation. In the second area, the article extends understanding of the performance proprieties of distributed representation, which opens the way for new applications.

HoloGN has its roots and inspiration in two specific models of the distributed associative memory: Sparse Distributed Memory (SDM) [46] and Hierarchical Graph Neuron. SDM is a mathematical model of human long-term memory, which is used for storing and retrieving large amounts (in the order of 2^{1000} bits) of information. HGN can be classified as a distributed associative memory in the sense that it models a stimulus as a graph of activities of multiple elementary neurons.

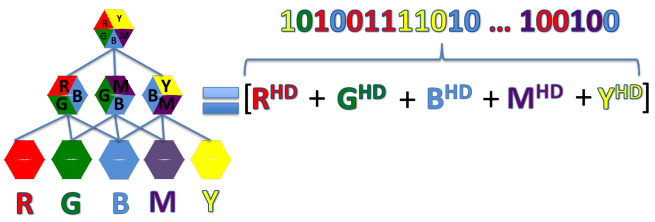


Figure 4.3: A high level illustration of the HoloGN: Representation of the Hierarchical Graph Neuron using Vector Symbolic Architecture; where the letters depict: R - red; G - green; B - blue; M - magenta; Y - yellow.

An important issue in HGN is the resource requirement overhead, specifically with regard to the number of processing elements required. This paper proposes a holographic approach, which (1) borrows the abstraction of the Graph Neuron (GN); and (2) enables a flat GN array to operate with higher level of accuracy and comparable recall time to that of HGN, without the need for a complex topology and additional nodes. The high-level logic of the proposed solution is presented in Fig. 3, which illustrates the ideas of both approaches. On the left HGN creates the representation of a pattern through communication indices of the activated GN-elements between neighbors. Thus, on the highest level HGN forms a representation of the whole pattern (illustrated with mixed colors in the figure). In contrast, on the right HoloGN achieves a similar result by encoding each GN element and the combination of their particular activations into a distributed representation.

In the following example the accuracy of the HoloGN recall was compared to the performance of the original HGN approach. For the sake of fair comparison the 7 by 5 pixels letters of the Roman alphabet (as in [45]) were used. In the memorization phase a set of noise-free images of letters was presented to both architectures. In the recall phase images of the same letters distorted with different levels of random distortions (between 1 bit corresponding to a distortion of 2.9% of the pattern's size and 5 bits equivalent to 14.3% distortion) were presented to the architectures for recall. In the case of HoloGN the pattern with the lowest normalized Hamming distance to the presented distorted pattern was returned as the output.

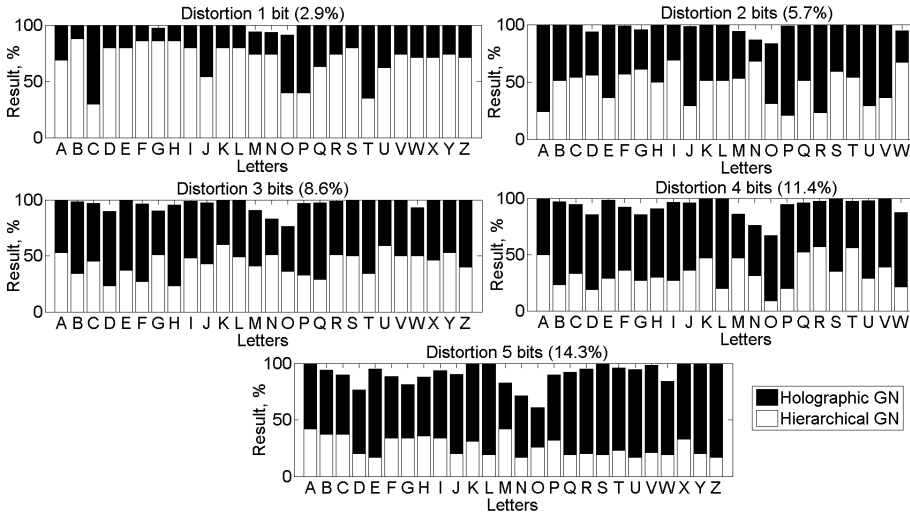


Figure 4.4: Results from testing black and white images of letters using recall patterns with distortions ranging from 1 bit (2.9%) to 5 bits (14.3%).

Fig. 4.4 presents the results of the accuracy comparison between the recall results for the HoloGN approach and the reference HGN architecture. To obtain the results 1000 distorted images of each letter for every level of distortion were presented for recall. The charts show the percentage of the correct recall output. The analysis shows that the performance of the HoloGN-based associative memory at least matches that of the original approach. However, in certain characters the recall is inferior to other letters. For example, the accuracy of the character "O" recognition is persistently lower than other letters. This is due its similarity to several other characters. In particular, when recalling "O" distorted by 5 bits HoloGN recall scoring is "C" - 5.2%, "D" - 11.4%, "G" - 11.9%, "Q" - 5.1%. In the simulations the average accuracy of HoloGN when recalling patterns is 51.7% higher than the accuracy of HGN.

There is a class of pattern recognition applications which requires an understanding

of the details of the recall results. For example, when a recall returns several possible patterns of given recall accuracy, the task would be to understand the overlapping elements. Given the rather conservative limits on the number of robustly decodable elements in a distributed representation it is important that the proposed HoloGN architecture can estimate the similarity between different patterns *without* decoding them. It is shown that the Hamming distance can act as a quantitative metric of the similarity via direct comparison of distributed representations, without the need for decoding those representations.

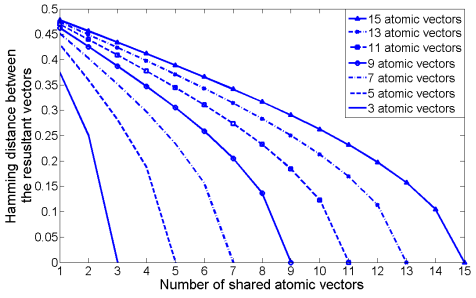


Figure 4.5: The normalized Hamming distance between two resulting vectors against number of components in common. The number of atomic vectors is the same.

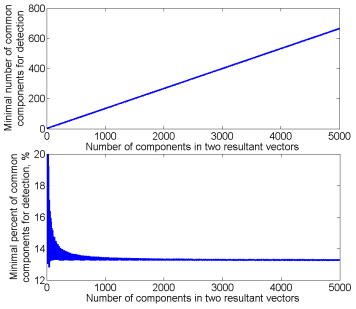


Figure 4.6: Minimal number of common components which are decodable between two patterns of the same size against the size of patterns, $d = 10000$, thr $= 10^{-6}$.

Figure 4.5 shows the normalized Hamming distances between two resulting vectors for different numbers of overlapping vectors. The results show that the larger the number of common elements, the smaller the normalized Hamming distance between resulting vectors. This method opens a way towards constructing and analyzing patterns far beyond VSA's robustly decodable capacity. The problem with practical application of this method, however, comes with the rapid convergence of the normalized Hamming distance indicator to 0.5, making the difference between analyzable patterns indistinguishable as illustrated in Figure 4.5. For example, for HoloGN representations of patterns with 15 elements, sub-patterns of 3 overlapped elements are robustly detected, while patterns with fewer overlapped elements are indistinguishable. Thus, the minimal number of overlapped elements in two patterns which can be robustly detected using the normalized Hamming distance indicator is called the bundle's *sensitivity*. Figure 4.6 demonstrates the development of the sensitivity threshold with the number of elements in the compared patterns. The results show that the number of components for robust detection grows linearly with the size of the pattern. Patterns with more than 500 elements should contain at least 14% overlapped elements to be robustly detected by the proposed method.

4.3 Paper C: On bidirectional transitions between localist and distributed representations: The case of common substrings search using Vector Symbolic Architecture

The transitions between localist and distributed representations are normally done only during the encoding of concepts and their later recall from the item-memory. Usually once encoded, all operations e.g., generalization and reasoning, are performed in the domain of distributed representations only. There are, however, cases when the transition back to the localist representation is essential, for example when characterizing complex systems [47], e.g. like industrial processes. This article presents an encoding approach which allows seamless bidirectional transitions between localist and distributed representation domains. The approach is demonstrated on the example of using Vector Symbolic Architecture for solving the problem of longest common substring search, which is a typical task when processing strings of symbols.

In the proposed approach patterns are firstly represented in a distributed way. Hamming distance is used as a similarity metric between patterns' representations. The longest string is chosen as a baseline. The cyclic shift of the distributively represented shorter string is used to check all possible substring combinations in the localist representation. For each shift transformation the Hamming distance between the longest string and the shifted string is measured. The tuple (shift position, Hamming distance) for each shift value is stored in a list sorted in descending order. After the substring has been shifted along all the baseline's elements the minimal Hamming distance amongst all shifts is determined. This value indicates the presence of the highest number of letters in the same positions between the two strings when the shorter string is shifted on the corresponding number of positions. The last step is the extraction of common substrings. It happens in the localist domain by shifting the shortest string the required number of positions. The two strings are subtracted and the task now is to find the longest sequence of zeros, which is the mask for the common substring. Thus, the proposed approach consists of three phases: encoding the strings into VSA distributed data representation; searching the shift positions in which strings have letters in the same positions; and extraction of the common substrings in the localist representation.

The principle of the search procedure is explained on an example illustrated in Fig. 4.7. There are two input strings to compare, string $P1$="bull" and string $P2$="vocabulary". First the distributed representations $P1^{HD}$ and $P2^{HD}$ of the strings are formed. The longest sting is chosen as a baseline. The search is implemented using only two operations: the cyclic shift of the representation $P1^{HD}$ of the shortest string; and the calculation of the Hamming distance between the first shifted representation $P1^{HD*}$ and the second representation $P2^{HD}$. In total one needs to perform $L1 + L2 - 1$ shifts in order to account for all possible relative positions of two strings. In this example, when four shifts are applied, the Hamming distance between the two representations $P1^{HD*}$ and $P2^{HD}$ will be the smallest, as the two strings now contain the largest number of

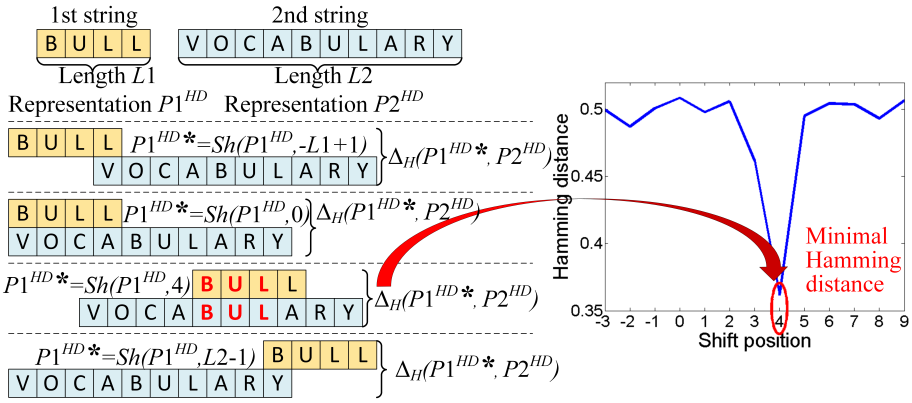


Figure 4.7: Illustration of search principles in the distributed domain.

overlapping characters.

Consider the case when hardware operating with HD-vectors is available and it can calculate operations such as cyclic shift, XOR, MAJORITY sum and Hamming distance in one clock cycle. With this assumption one can see that maximal size of the loop in the proposed approach equals $L1 + L2 - 1$, this statement is also relevant to the extraction of the common substring when the shift is known. Thus, the overall computational complexity of the proposed approach is $\Theta(L1 + L2 - 1)$, which is the same as the complexity of the suffix trees approach and better than for the standard approach using dynamic programming with $\Theta(L1 \cdot L2)$ operations.

The main limitations of the proposed approach stem from the nature of the distributed representation. The restriction of the approach is its sensitivity i.e., the minimal length of the common substring, which can be robustly detected. The simulation results give a practical restriction: in order to be robustly detected the length of the common substring should be more than 4% of the longest string.

4.4 Paper D: Fault Detection in the Hyperspace: Towards Intelligent Automation Systems

Time condition monitoring and intelligent maintenance of industrial systems are becoming increasingly important functions of automation systems, with the growing application of artificial intelligence methods. This article considers a biologically inspired approach for fault detection and identification in generic complex systems of systems. The approach adopts a mathematical framework previously used for the modeling of human cognitive abilities called Vector Symbolic Architectures [48]. In simple words the proposed approach aims to learn and generalize temporal patterns in streams of telemetric data generated across the plant during its stable and/or faulty operation scenarios. In this way the created on-line model is continuously updated as the particular plant or automation system evolves. To this end the proposed approach is suggested as an overlay layer on top of the existing fault detection and management systems, which would enhance the robustness of decision making of the overall system.

In the proposed approach the inference of the potential fault is done by a central unit, which collects states from all N components. For this patterns of the system-wide state corresponding to different faults are stored in the unit. In order to construct the pattern the unit measures components states at the moment of the pattern construction. The core of the proposed approach is Holographic Graph Neuron [49] – a one-shot learning associative memory for pattern recognition.

The proposed approach functions in two phases: the learning phase and the fault identification phase. The learning phase could either be performed off-line and using system-level simulation facilities or in real time, learning the fault situation through an interaction with a SCADA system and a human operator. The result of the learning phase is a collection of patterns of system-wide state corresponding to the particular fault encoded using the HoloGN approach. In the fault identification, each component first informs the central unit of its current state. The pattern of states is then encoded using HoloGN. Finally the recall operation is performed on the collection of patterns from learning phase. The result of the recall is a list of potential faults.

The case study used for demonstrating the proposed cognitive fault detection system is a generic nuclear power plant model provided by Fortum Power and Heat, a power utility with nuclear power plant operation license in Finland. The Apros 6 process simulator was used to run the model. The functional failure identification and propagation framework was used to analyze 116 automation components as the sources of hardware faults [50]. Most of these components were pump and valve actuators. From all possible faults, 92 faults actually affected the steady state operation of the power plant model and thus can be detected by a data driven fault detection system. In the case study the model was driven to 11 power production levels in order to get more simulation data for the set of faults. The 92 faults, which are detectable in the steady state of the plant, are simulated for every one of the 11 power levels (1012 simulations in total). The results of these simulations were used to build data sets for training and testing the fault detection systems.

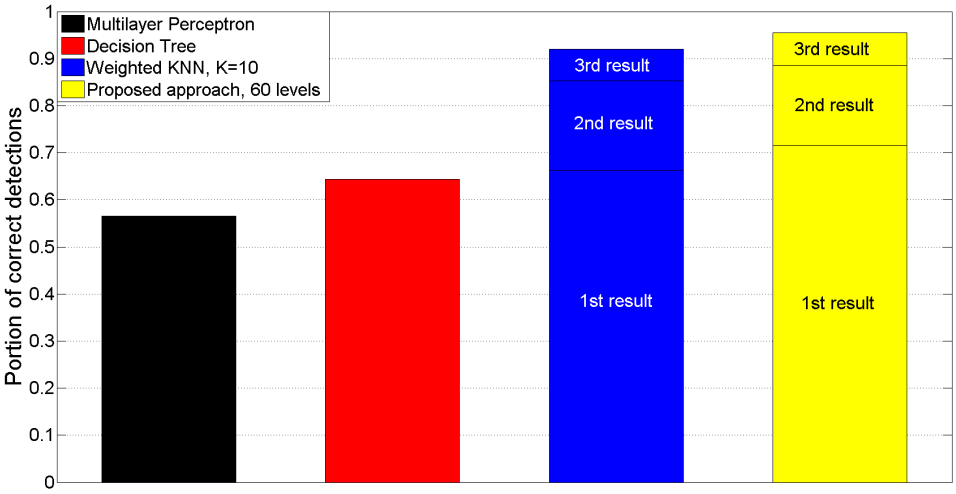


Figure 4.8: The results of benchmarking: ANN, decision tree, KNN and the proposed approach.

The performance of the proposed approach was compared to the performance of a multilayer perceptron Artificial Neural Network (ANN), a decision tree, and a K-nearest-neighbors (KNN) classifier. The accuracy was measured as the ratio of the correctly identified faults over all presented cases from the test set. The results of comparison of the considered approaches are shown in Fig. 4.8. The main result indicated by the figure is that the accuracy of the fault identification by the proposed approach outperforms the other approaches even when considering the top result. The best accuracy was demonstrated by both KNN classifier and the proposed approach (0.68 and 0.71 correspondingly). Showing either matching or in most of the cases superior performance over the traditional approaches our approach has one more unique advantage: it is suitable for distributed implementation. Note, however, that the proposed approach does not necessarily always outperform the benchmarked techniques, e.g. ANN is very powerful for classification when borders between classes are non-linear.

Chapter 5

Conclusions and Future Work

"But at least I tried."

Randle Patrick McMurphy
in "One Flew Over the Cuckoo's Nest"

The aim of this work is to explore the capabilities of biologically inspired approaches to pattern recognition problems in different domains. Specifically, this thesis sets the goal of investigating the applicability of Vector Symbolic Architectures for pattern processing and exploring its potential application areas. Among the key properties of VSAs are estimation of gradual similarity, efficient use of representational resources, its suitability for usage in associative memories allowing a natural generalization, graceful degradation of representation, resistance to noise and uncertainty, learning from examples, and analogical reasoning. The hypothesis in this work is that VSAs' properties would facilitate progress in pattern recognition problems. The results in the appended papers demonstrate on par performance with traditional methods for the fault detection and longest common substrings search problems.

5.1 Contributions

Based on the results provided in the appended papers the following answers can be stated for the research questions posed in Chapter 1:
Q1 *Can VSAs be applied for representation, learning and subsequent classification of temporal patterns in a showcase of a classification of vibration sensors measurements from vehicles?*
It was shown in Paper A that the classification of temporal patterns can be done based on adopting Vector Symbolic Architectures for representation of continuous signals. The approach was exemplified for vehicle classification for Intelligent Transportation Systems using measurements of passing vehicles from vibrations sensors. The proposed VSA-based classifier is capable of storing distributed representations of up to 100 samples when building it using HD-vectors of size 1kB.
Q2 *Can a VSA-based architecture be capable of forming and memorizing distributed rep-*

resentations for patterns of generic sensor stimuli be created?
Paper B presented the VSAs based architecture for the storage and the recall of patterns of generic sensor stimuli. The architecture is built upon the state of the art algorithm called Hierarchical Graph Neuron. The adoption of the Vector Symbolic representation ensures a single-layer design for the approach, which leads to much simpler computational operations. The noise resistance properties were improved thanks to usage of distributed representations leading to substantial improvement of pattern recall accuracy.
Q3 *Can the computational task of the longest common search benefit from the usage of distributed representations?*
Paper C used the proposed architecture for solving the problem of finding common substrings. The complexity of the proposed approach for the longest common substring search in terms of vector operations is comparable with the best traditional algorithms. The simulation results provide a practical restriction to the approach that for robust detection the length of the common substring should be more than 4% of the longest string.
Q4 *Can such architecture be successfully applied for fault detection in generic complex systems of systems?*
The results of the case-study of fault classification using the generic nuclear power plant model in Paper D showed that the architecture performed on a par in terms of the the fault detection accuracy with the benchmarked machine learning techniques.

5.2 Future Work

The results presented in this thesis are promising, but there are still many questions and research directions left to be explored. Paper A presented a possible encoding scheme for continuous signals. However, it is brittle in the sense that the approach relies on the extraction of features from the signals. The question of the direct mapping of continuous signals into HD-vectors while maintaining the relevant properties of signals in distributed representations requires in-depth study. The emergent methods of featureless similarity estimation e.g., data smashing [51, 52] can serve as a basis for inspiration.

The effects of different types of noise on noise resistance properties of the proposed architecture for formation of distributed representations of generic sensor stimuli should be investigated. Furthermore, the current design of the architecture is still centralized. The distributed version of the architecture is most likely to fulfill the flexibility and the decentralization requirements of future automation and manufacturing system. The performance of the distributed system could be studied in the same scenario as in Paper D, thus allowing a comparison between the distributed version and several centralized methods. Furthermore, recent works on pattern prediction with VSAs [41, 42] demonstrated that the usage of coefficients regulating the contribution of different pattern's dimensions (e.g., using mutual information) improves system's performance. The current version of the architecture in Paper B does not consider usage of weighting coefficients. However, the usage of weighting coefficients could improve the architecture's performance in some applications. New applications areas for the usage of the proposed methods should be explored. One of the promising directions is analysis of biomedical signals.

The properties of the approach with distributed representation for the longest common substring search demand exploration of its applicability for the task of alignment of DNA's reads. Potentially the vector operations can be efficiently implemented using powerful graphic accelerators which can lead to better performance compared to the current implementation of the alignment task.

Finally, in the appended papers a simple matrix based storage of codewords was used. However, usage of associative memories, e.g. Sparse Distributed Memory [46] or Hebb's cell assemblies [29] may provide the architecture with a mechanism for emerging hierarchies of generalization.

References

[1] T. van Gelder, "Distributed vs. Local Representation," in *The MIT Encyclopedia of the Cognitive Sciences*, R. Wilson and F. Keil, Eds. MIT Press, 1999, pp. 235–237.

[2] T. Plate, *Distributed Representations and Nested Compositional Structure.* University of Toronto, PhD Thesis, 1994.

[3] ——, *Holographic Reduced Representations.* CLSI Publications, 2003.

[4] G. Hinton, J. McClelland, and D. Rumelhart, "Distributed Representations," in *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1. Foundations*, D. Rumelhart and J. McClelland, Eds. MIT Press, 1986, pp. 77–109.

[5] G. Hinton, "Mapping Part-Whole Hierarchies into Connectionist Networks," *Artificial Intelligence*, vol. 46, pp. 47–75, 1990.

[6] P. Smolensky, "Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems," *Artificial Intelligence*, vol. 46, pp. 159–216, 1990.

[7] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[8] R. Gayler, "Vector Symbolic Architectures Answer Jackendoff's Challenges for Cognitive Neuroscience," in *Proceedings of the Joint International Conference on Cognitive Science. ICCS/ASCS*, , Ed. , 2003, pp. 133–138.

[9] S. Levy, "Vector symbolic architectures," http://home.wlu.edu/~levys/vsa.html#.

[10] J. Fodor and Z. Pylyshyn, "Connectionism and cognitive architecture. A critical analysis," *Cognition*, vol. 28, no. 1-2, pp. 3–71, 1988.

[11] R. Jackendoff, *Foundations of language. Brain, Meaning, Grammar, Evolution.* Oxford University Press, 2002.

[12] P. Kanerva, J. Kristofersson, and A. Holst, "Random Indexing of text samples for Latent Semantic Analysis," in *Proceedings of the 22nd Annual Conference on the Cognitive Science Society*, 2000, p. 1036.

[13] M. Sahlgren, "An Introduction to Random Indexing," in *Proceedings of the 7th International Conference on Terminology and Knowledge Engineering*, 2005, pp. 1–9.

[14] C. Eliasmith, T. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[15] C. Eliasmith, *How to Build a Brain. A Neural Architecture for Biological Cognition.* Oxford University Press, 2013.

[16] P. Kanerva, "The Spatter Code for encoding concepts at many levels," in *ICANN'94, Proceedings of the International Conference on Artificial Neural Networks*, Springer-Verlag, Ed. , 1994, pp. 226–229.

[17] ——, "A Family of Binary Spatter Codes," in *ICANN'95, Proceedings of the International Conference on Artificial Neural Networks*, , Ed. , 1995, pp. 517–522.

[18] ——, "Binary Spatter-Coding of ordered K-tuples," in *ICANN'96, Proceedings of the International Conference on Artificial Neural Networks*, ser. Lecture Notes in Computer Science, , Ed., vol. 1112. Springer, 1996, pp. 869–873.

[19] ——, "Fully Distributed Representation," in *Proceedings of 1997 Real World Computing Symposium, RWC'97*, , Ed. , 1997, pp. 358–365.

[20] ——, "Dual role of analogy in the design of a cognitive computer," in *Advances in Analogy Research. Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*, D. Gentner and K.J. Holyoak and B. Kokinov, Ed. , 1998, pp. 164–170.

[21] ——, "Computing with large random patterns," in *The Foundations of Real-World Intelligence*, Y. Uesaka and P. Kanerva and H. Asoh, Ed. , 2001, pp. 251–272.

[22] ——, "What We Mean When We Say What's the Dollar of Mexico?" in *AAAI Fall Symposium. Quantum Informatics for Cognitive, Social, and Semantic Processes*, , Ed. , 2010, pp. 2–6.

[23] ——, "Pattern completion with distributed representation," in *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, , Ed., vol. 2. , 1998, pp. 1416–1421.

[24] ——, "Large patterns make great symbols. An example of learning from example," in *Hybrid Neural Systems*, ser. Lecture Notes in Computer Science, S. Wermter and R. Sun, Ed., vol. 1778. Springer, 2000, pp. 194–203.

[25] ——, "Computing with 10,000-bit words," in *Proceedings of 52nd Annual Allerton Conference on Communication, Control, and Computing*, , Ed. , 2014, pp. 1–7.

[26] ——, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, October 2009.

[27] D. Rachkovskij, I. Misuno, and E. Revunova, "Random Vector Indexing of documents and Semantic Representations of words," in *Proceedings of 5th All-Russian Science and Technology Conference Neuroinformatics-2003*, 2003, pp. 213–218.

[28] S. Gallant, "Context Vectors. A Step Toward a Grand Unified Representation," in *Hybrid Neural Systems*, ser. Lecture Notes in Computer Science, S. Wermter and R. Sun, Eds., vol. 1778. Springer, 2000, pp. 204–210.

[29] D. Rachkovskij, E. Kussul, and T. Baidyk, "Building a world model with structure-sensitive sparse binary distributed representations," *Biologically Inspired Cognitive Architectures*, vol. 3, pp. 64–86, 2013.

[30] I. Misuno, D. Rachkovskij, and S. Slipchenko, "Vector and distributed representations reflecting semantic relatedness of words," *Mathematical Machines and Systems*, vol. 3, pp. 50–66, 2005. (In Russian).

[31] I. Misuno, D. Rachkovskij, S. Slipchenko, and A. Sokolov, "Processing text information with vector representations," in *International Workshop on Inductive Modelling '05 (IWIM-05)*, 2005, pp. 230–236. (In Russian).

[32] I. Misuno, D. Rachkovskij, A. Sokolov, and S. Slipchenko, "Searching for text information with vector representations," *Problems in Programming*, vol. 4, pp. 50–59, 2005. (In Russian).

[33] I. Misuno, D. Rachkovskij, and S. Slipchenko, "Distributed representation of data in classification tasks," *System Technologies*, vol. 42, no. 1, pp. 107–113, 2006. (In Russian).

[34] D. Rachkovskij, "Linear classifiers based on binary distributed representations," *International Journal Information Theories and Applications*, vol. 14, no. 3, pp. 270–274, 2007.

[35] S. Slipchenko, I. Misuno, and D. Rachkovskij, "Properties of coarse coding with random hyperrectangle receptive fields," *Mathematical Machines and Systems*, vol. 4, pp. 15–29, 2005. (In Russian).

[36] M. Jones and D. Mewhort, "Representing Word Meaning and Order Information in a Composite Holographic Lexicon," *Psychological Review*, vol. 114, no. 1, pp. 1–37, 2007.

[37] M. Sahlgren, A. Holst, and P. Kanerva, "Permutations as a Means to Encode Order in Word Space," in *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*, 2008, pp. 1300–1305.

[38] G. Recchia, M. Jones, M. Sahlgren, and P. Kanerva, "Encoding Sequential Information in Vector Space Models of Semantics Comparing Holographic Reduced Representation and Random Permutation," in *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*, 2010, pp. 865–870.

[39] G. Recchia, M. Sahlgren, and P. K. M. Jones, "Encoding Sequential Information in Semantic Space Models. Comparing Holographic Reduced Representation and Random Permutation," *Computational Intelligence and Neuroscience*, pp. 1–18, 2015.

[40] A. Joshi, J. Halseth, and P. Kanerva, "Language Recognition using Random Indexing," Cornell University Library, pp. 1–7, 2015, http://arxiv.org/abs/1412.7026.

[41] O. Rasanen and S. Kakouros, "Modeling Dependencies in Multiple Parallel Data Streams with Hyperdimensional Computing," *Signal Processing Letters, IEEE*, vol. 21, no. 7, pp. 899–903, 2014.

[42] O. Rasanen and J. Saarinen, "Sequence Prediction With Sparse Distributed Hyperdimensional Coding Applied to the Analysis of Mobile Phone Use Patterns," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–12, 2015.

[43] O. Rasanen, "Generating Hyperdimensional Distributed Representations from Continuous Valued Multivariate Sensory Input," in *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, 2015, pp. 1943–1948.

[44] S. Levy, S. Bajracharya, and R. Gayler, "Learning behavior hierarchies via high-dimensional sensor projection."

[45] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 212–229, February 2008.

[46] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, 1988.

[47] B. Emruli, F. Sandin, and J. Delsing, "Vector space architecture for emergent interoperability of systems by learning from demonstration," *Biologically Inspired Cognitive Architectures*, vol. 11, pp. 53–64, 2015.

[48] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Computation*, vol. 25, no. 8, pp. 2038–2078, 2013.

[49] E. Osipov, A. I. Khan, and A. Anang, "Holographic graph neuron," in *Computer and Information Sciences (ICCOINS), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[50] N. Papakonstantinou, S. Proper, B. O'Halloran, and I. Y. Tumer, "Simulation Based Machine Learning For Fault Detection In Complex Systems Using The Functional

Failure Identification And Propagation Framework," in *ASME CIE, Buffalo NY, USA*, 2014, pp. 1–10.

[51] H. Lipson and I. Chattopadhyay, "Data smashing," in *Proceedings of The Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, 2014, pp. 7–14.

[52] I. Chattopadhyay and H. Lipson, "Data smashing. uncovering lurking order in data," *Journal of the Royal Society Interface*, vol. 11, no. 101, pp. 1–11, 2015.

Part II

Paper A

Brain-like classifier of temporal patterns

Authors:
Denis Kleyko and Evgeny Osipov

Reformatted version of paper originally published in:
IEEE International Conference on Computer and Information Sciences (ICCOINS), 2014, Kuala Lumpur, Malaysia.

Brain-like classifier of temporal patterns

Denis Kleyko and Evgeny Osipov

Abstract

In this article we present a pattern classification system which uses Vector Symbolic Architecture (VSA) for representation, learning and subsequent classification of patterns, as a showcase we have used classification of vibration sensors measurements to vehicles types. On the quantitative side the proposed classifier requires only 1 kB of memory to classify an incoming signal against of several hundred of training samples. The classification operation into N types requires only $2 * N + 1$ arithmetic operations this makes the proposed classifier feasible for implementation on a low-end sensor nodes. The main contribution of this article is the proposed methodology for representing temporal patterns with distributed representation and VSA-based classifier.

1 Introduction

The problem of pattern classification is not new and appears in many areas ranging from semantic analysis of natural languages to intelligent transportation systems. During past years vast number of methods based on statistical analysis, neural network and others has been established [1]. Many of these methods require the presence of the entire sample set, manual time-consuming engineering of features and in many cases having processing of the data set. Our long term aim is to avoid these time consuming processes as much as possible. In this article we propose a brain-like classification system of time-sequences of raw sensory data.

Vector Symbolic Architecture is a bio-inspired method for representing semantically bound information. Similarly to brain activity where simple mental events involve the simultaneous activity of very large number of dispersed neurons [2] the data in VSA is represented distributively. Where a single concept is associated with multiple codes. In VSA this is achieved by using codewords of very large dimension. In this article we utilize a subclass of VSA based on so-called Binary Spatter Codes (BSC) [2]. A codeword in BSC is randomly generated binary vector of very high dimension, i.e. several thousand bits. In VSA the structured information is represented by meshing up different codes using simple arithmetical operations: bit-wise exclusive OR, different types of summation. In a way under distributed representation a data structure can be seen as a scalar of codeword's dimension. The unique feature of VSA is its ability to learn and generalize. So far VSA was mainly used in the context of semantic analysis of natural languages.

In this article we demonstrate the capability of VSA to represent and classify temporal patterns. By temporal patterns we understand signals from different types of sensors

in response to external events featuring repeated patterns in time. For example a vehicle passage generates vibration sequences unique to different types of vehicles another example on a longer time scale would be measurements of typical temperature changes in a specific environment. The main contribution of this article is the proposed methodology illustrated in Fig. 1 for encoding signal patterns using distributed representation and VSA based classifier. On the quantitative side the proposed classifier requires only 1 kB of memory to classify an incoming signal against of several hundred of training samples. To classify into N types the classification operation requires only $2*N+1$ arithmetic operations this makes the proposed classifier feasible for implementation on low-end sensor nodes.

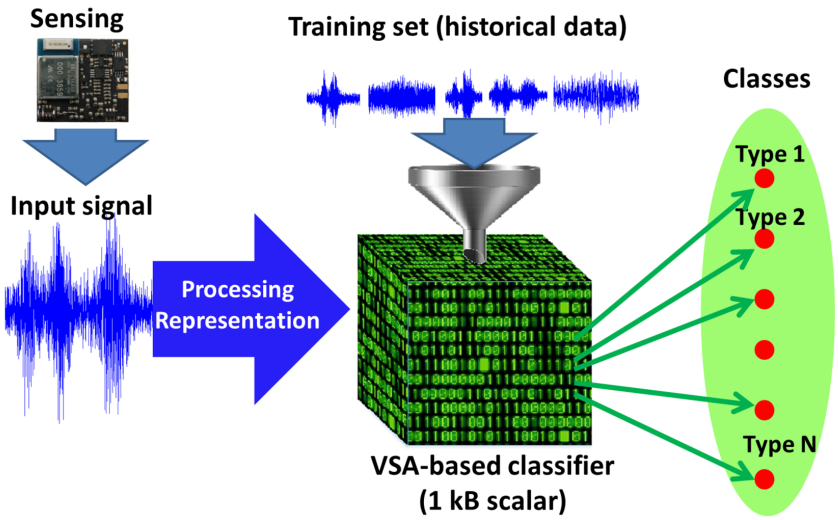


Figure 1: System architecture.

The article is structured as follows. The related work follows in Section 2. Section 3 presents the fundamentals of the theory of Vector Symbolic Architecture and Binary Spatter Codes relevant to the scope of the article. The problem formulation and showcase description are demonstrated in Section 4. Section 5 presents the main contribution of the article - the VSA-based classifier. The discussion is presented in Section 6. We conclude the article in Section 6.

2 Related work

Most of the classic methods for pattern recognition rely on manual definition of features to look for. The obvious limitation is reliance on expert knowledge and as a result long design phase of the classifier. On the other hand classifiers can also be built based on neural networks, which do not require initial domain knowledge [3], however defining classification rules (rules extraction) is difficult [4]. Amongst the alternative methods for pattern analysis and classification works [5], [6] are of particular relevance to the approach presented in this article.

The methodology proposed in [5] is in the domain of visual image recognition. In essence it constructs a distributed representation of all possible transforms of the particular object for further fast and reliable image recognition.

Hierarchical Graph Neuron [6] is an approach for memorizing of patterns of generic sensor stimuli for later template matching. In contrast to contemporary machine learning approaches, GN allows induction of new patterns in the learning set without the need for retraining. A certain degree of similarity exists in hierarchical structure for maintaining pattern's internal cross correlation information. This property is native to distributed representation in general and VSA in particular.

Vector Symbolic Architectures (VSA) [7] is a class of connectionist models that use hyper-dimensional vectors to encode structured information as distributed or holographic representation. In this technique structured data is represented by performing basic arithmetic operations on field-value tuples. Distributed representations of data structures is an approach actively used in the area of cognitive computing for representing and reasoning upon semantically bound information [8], [9], [10]. In [11] a VSA-based knowledge-representation architecture is proposed for learning arbitrarily complex, hierarchical, symbolic relationships (patterns) between sensors and actuators in robotics. Recently the theory of hyper-dimensional computing and VSA in particular were adopted for implementing novel communication protocols and architectures for collective communications in machine-to-machine communication scenarios including wireless sensor networks [12], [13]. The first work demonstrates unique reliability and timing properties essential in the context of industrial machine-to-machine communications. The latter work shows the feasibility of implementing collective communications using current radio technology.

3 Fundamentals of Vector Symbolic Architecture and Binary Spatter Codes

Distributed representation of data structures is an approach actively used in the area of cognitive computing for representing and reasoning upon semantically bound information [2], [14]. While in conventional computing architectures the entities are represented by single units (e.g. a field "Name" in relation databases has a predefined offset amongst other fields and name "John" has a unique representation in ASCII codes), in distributed

representation all entities are represented by random vectors of binary values of very high dimension (HD-vectors).

High dimensionality means here several thousand of binary positions for representing a single entity. In [2] it is proposed to use vectors of 10000 binary elements. In this article for computational convenience we will use 8196 bits or 1 kB vectors.

Randomness means that the values on each position of an HD-vector are independent of each other, and "0" and "1" components are equally probable, $p_0 = p_1 = 0.5$. On very high dimensions, the distances from any arbitrary chosen HD-vector (in other words point in hyperdimensional space) more than 99.99 % of all other vectors in the representation space are concentrated around 0.5 Hamming distance. Interested readers are referred to [2] and [15] for comprehensive analysis of the hyperdimensional space.

Binary vectors with these properties could be generated based on Zadoff-Chu sequences [16] widely used in telecommunications to generate pseudo-orthogonal preambles. Using this principle a sequence of K pseudo-orthogonal vectors to a given initial random HD-vector A is obtained by cyclicly shifting vector A on $1 < i \leq K$. Further in the article we denote this operation as $Sh(A, i)$. Cyclic shift operation has the following properties: cyclic shift is invertible, i.e. if $B = Sh(A, i)$ then $A = Sh(B, -i)$; it preserves distance; product is dissimilar to the vector being shifted.

In Vector Symbolic Architectures structures IDs of fields in a structure and their values are encoded using different HD-vectors. The following operations are used for structure composition. Binding, i.e. assigning a value to a field is implemented by bit-wise XOR operation on corresponding vectors further denoted as $\otimes$. The important properties of XOR operation are: binding is invertible, i.e. if $C = A \otimes B$ then $C \otimes A = B$; binding distributes over bundling or addition, i.e. $C \otimes (A \oplus B) = C \oplus A \otimes C \oplus B$; binding preserves distance; product is dissimilar to the vectors being binded.

Joining several field-value tuples into a structure is done by bundling operation, implemented by thresholded sum of the HD-vectors representing tuples. A bit-wise thresholded sum of n vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. Further we use terms "thresholded sum" and "MAJORITY sum" interchangeably and denote them as $[A + B + C]$. The relevant properties of the MAJORITY sum are: the result is a random vector, i.e. the number of "1" components is equal to the number of "0" components; the result is similar to all vectors included in the sum; number of vectors involved into MAJORITY sum must be odd.

The extraction of information out of VSA represented structures is the reverse to the encoding process. Namely, a particular vector is extracted by XORing the HD-vector representing the compositional structure.

Consider the following example for better clarity. Suppose the following VSA is used to encode a structure with two fields: $S = (F_1 \otimes V_1) \oplus (F_2 \otimes V_2)$. To decode value V_1 the following set of operations is performed on S:

1. XOR with HD-vector representing role F_1 in order to retrieve its filler: $V_1' = S' \otimes F_1 = V_1 \oplus NOISE$.

2. Pass noisy vector V_1' to the item memory for finding the best clean match, i.e.

search for an entry with minimal to V_1' Hamming distance.

The distributed representation has important capability to generalize based on previous examples. However in the scope of this article we do not utilize this property and concentrate on using VSA for an efficient storing and fast patterns classification.

4 Problem statement, solution overview and the showcase scenario

Consider generic signal exhibiting a certain pattern in time illustrated in Fig. 2. The task of raw sensory signal conversion into VSA represented pattern is formulated as finding and quantifying distinct changes in signal level and representing them using HD-vectors. For example the signal in Fig. 2 features changes at 9 positions. The first and the last positions correspond to the level of the ambient noise experienced by the sensor in the absence of external stimuli. The task is to extract a *set* of quantified significant deviations of the signal from the noise floor. The problem of detecting the magnitude of the signal's change is outside the scope of this article since it rather falls in the domain of signal processing. Number of traditional algorithms can be used for detecting significant signal changes. Summarizing its functionality, as the output the algorithm should produce a *series* of values for magnitudes of *significant* signal's changes. The values of magnitudes are quantized into a finite number of discrete quantization levels as illustrated in Fig. 2. The number of the quantization levels is application domain specific and can be selected automatically without manual pre-engineering. For a collection of signals having similar underlying generating stochastic process the obtained from the algorithm series of change-magnitude values form patterns, which are similar across specific types of the signals' collection. Thus the signal examplified in Fig. 2 features pattern: 1-2-1-4-1-2-1.

This article addresses two problems:

1. Vector Symbolic Architectures representation of patterns given as an output from a signal processing algorithm in the form of quantized values of changes' magnitude.

2. Given a set of VSA-represented patterns construct a classifier for fast classification of the incoming digitized sequences.

4.1 Solution overview

Despite the fact that VSA data representation has an embedded learning capability as described in Section 3 in this article we seek for a simpler solution. We intend to demonstrate the capability of VSA-based data representation as a memory efficient storage of historical data and that it feasible to construct an accurate and fast classifier based upon this structure.

Essentially the overall idea with our solution is: 1.) to take the detected pattern of significant signal changes; 2.) represent each pattern as a VSA-based structure; 3.)

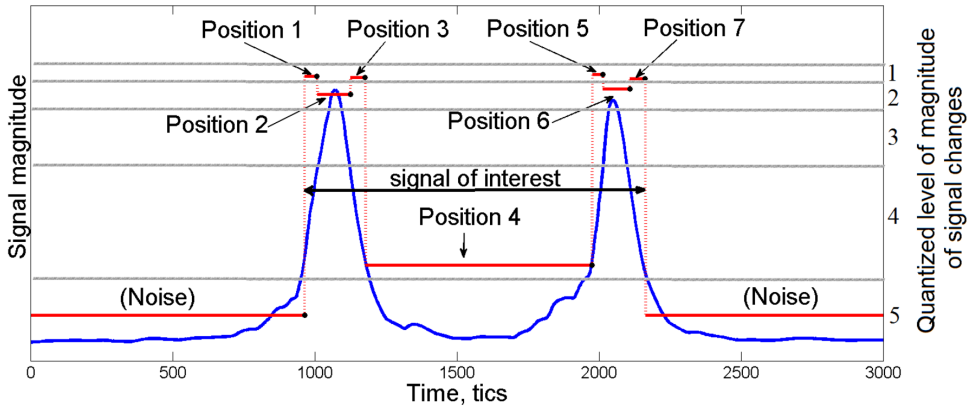


Figure 2: An example of signal ready for VSA representation. This signal features pattern 1-2-1-4-1-2-1.

combine many such structures into a single VSA-based representation, which will further be used for the classification operation.

The latter step was previously introduced in [11] for VSA-based control of robot actions. Our solution extends this idea to classification of a general class of sensory signals experiencing temporal patterns.

4.2 The showcase scenario

The solution presented in this article has a real-life show case scenario, which we use to demonstrate the performance of the proposed architecture. At our disposal we have raw measurements of vehicle passages collected from a vibration sensor installed on the road surface in the university's testbed of Intelligent Transportation System. Typical measurements for different classes of monitored vehicles are demonstrated in Fig. 3. The blue colored curves show the raw signal while the red curves show the filtered signal, which we will use for further analysis and VSA-based representation. In total we possess 46 samples of signal collected from measuring passages of 30 passenger cars, 10 three-axles trucks and six two-axles trucks.

The patterns observed as the output from signal change detection algorithm, which are the input to our VSA-based classifier is illustrated Tab. 1. For the space-saving reasons we cut the number of processed patterns for passenger cars to 11. Two remarks are worth making at this point. First, one can clearly see typical patterns repeating for vehicles of the same class. Although we do not discuss the topic of using learning capabilities of VSA-based representation for generalization of the training set, this observation indicates potential feasibility for doing so.

Second, it is important to note that the detected patterns are of different length and a logical question is whether they are comparable at all. Here we call the reader to

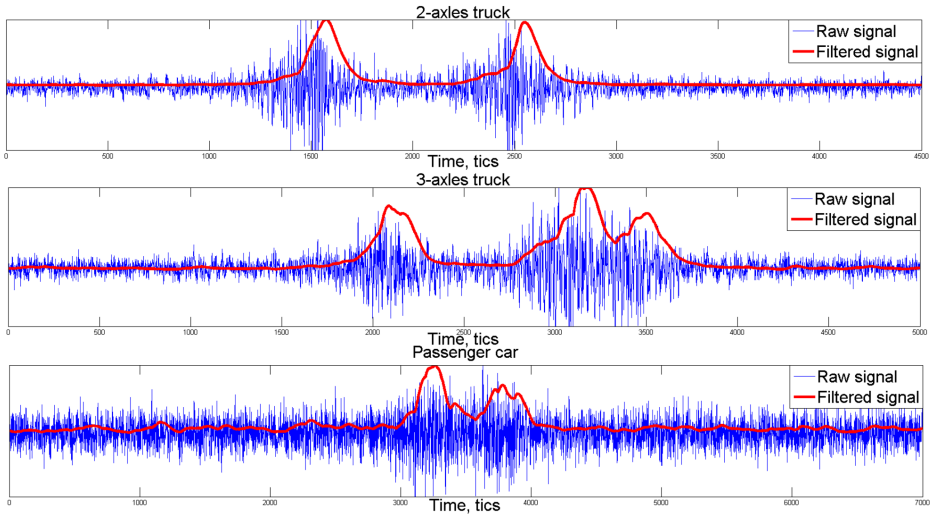


Figure 3: Examples of raw and filtered signals for 2-axles and 3-axles trucks and a passenger car respectively.

remember the distributed nature of the VSA representation. Namely, in VSA encoding it does not matter the relative order of the pattern's features since each encoded component of the pattern will be interleaved with all others during the binding and bundling operations. Remembering this, may and will use the position of each detected element of the pattern for the encoding purposes without loosing the sense of comparison.

5 Constructing the VSA-based classifier

The input to the the procedure of constructing the VSA-based classifier is the detected patterns of significant signal changes from all collected samples. The stages for constructing a VSA based classifier are: 1.) encoding of patterns into a VSA representation; and 2.) bundling several VSA-represented patterns into a single VSA representation (HD-vector), which will serve as the classifier.

We build our classifier in form (1) following similar line of reasoning as in [11]. In (1) SUM is the operation of MAJORITY summation, P_j is a VSA-represented pattern of a signal, T_i is an HD vector representing the type to which the signal belongs to. In the case of our showcase scenario we have three types: a passenger car, a two-axles truck and a three-axles truck.

$$CLASSIFIER = SUM_{j=1..N} P_j \otimes T_i | \{i = 1..K\}. \quad (1)$$

Table 1: Patterns - output of the signal changes' detection algorithm per class. processing

No.	Pattern								
Patterns for 2-axles trucks									
1	1	2	1	4	1	2	1		
2	1	2	1	4	1	2	1		
3	1	2	1	4	1	2	1		
4	1	2	1	4	1				
5	3	4	1	2	1				
6	1	1	3	3	1	2	1		
Patterns for 3-axles trucks									
7	1	3	4	1	4	1			
8	1	1	2	1	4	1	1	3	
9	3	4	1	2	1	1	1		
10	3	4	1	3	1	3			
11	3	4	1	1	2	1	1		
12	1	1	1	4	3	3			
13	1	1	1	4	1	1	3	3	
14	3	4	3	3	1	1			
15	1	2	1	4	1	2	1	1	1
16	3	4	3	1	2	1			
Patterns for passenger cars									
17	1	2	1	2	1	1	1		
18	1	3	1	3	1	1	1		
19	1	3	1	1	1				
20	3	4	1	1	1				
21	1	1	3	1	1	1			
22	1	2	1	2	1	1	1		
23	1	2	1	2	1	1	1		
24	1	3	1	3	1	1	1		
25	1	1	3	1	3	1	1		
26	1	1	2	1	1	3			
27	1	1	1	2	1	3			

5.1 Encoding of patterns into VSA representation

Even a quick look on Tab. 1 reveals repeating quantization levels in each pattern. Moreover, some patterns, which belong to different classes partially overlap, as for example is the case for pattern 5 for the two-axles truck and pattern 9 representing the three-axles truck. The specifics of the distributed representation is such that if a structure is constructed by overlaying repeated elements they start to dominate in the resulting pattern. This, in simple words, makes VSA representations of different patterns look alike in terms of Hamming distance between each other. The main goal of the encoding

task is, therefore, to make the VSA representation of patterns which are different even in a single position mutually orthogonal.

Thus, we want to encode, for example, level 1 on position 1 and position 3 of the particular pattern by different HD-vectors. We solved this task by performing cyclic shift operation on the initial HD-vector assigned for a particular quantization level: $Sh(L_i, p_j)$. The first argument of the operation is the initial HD-vector which is used to encode quantization level L_i and the second argument is the relative position of this level from the beginning of the pattern. Recall that the shift operation generates another random vector orthogonal to the one being shifted.

Secondly, when constructing VSA representation of the particular pattern we XOR HD-vectors encoding individual elements. Recall that each XOR operation produces a random vector, which is dissimilar to the all HD-vectors-components. Visually one could imagine that the result of the XOR operation is located very far away in the representation space from all the XOR'ed HD points. Schematically the idea for encoding a pattern into VSA representation is illustrated in Fig. 4 where a blue circle is 2D projection of the hyper-dimensional space.

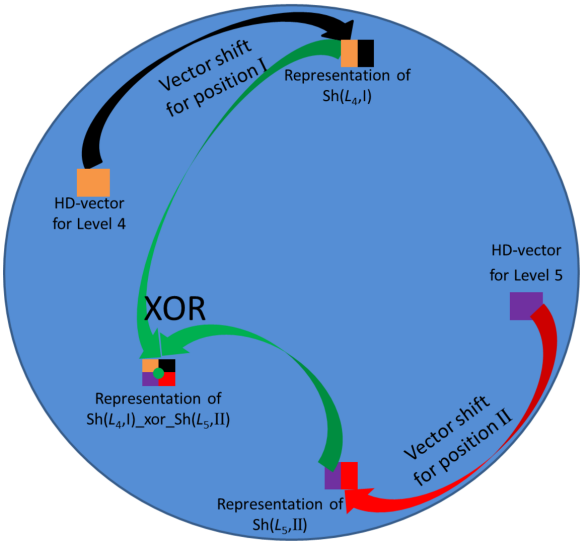


Figure 4: Illustration of the idea for VSA encoding of a pattern by cyclic shift and XORing of individual elements.

As an example of the encoding scheme consider pattern 4 describing a passage of the two-axles truck (1-2-1-4-1). The pattern is represented by VSA which is constructed as follows:

Table 2: Hamming distance for probing results with the classifier

No.	1	2	3	4	5	6	7	8	9
2-x	**.44**	**.44**	**.44**	**.44**	**.45**	**.43**	.50	.50	.50
3-x	.50	.50	.50	.50	.50	.50	**.44**	**.43**	**.44**
pass	.49	.49	.49	.50	.50	.51	.51	.49	.50
No.	10	11	12	13	14	15	16	17	18
2-x	.50	.49	.50	.51	.49	.49	.50	.50	.50
3-x	**.44**	**.44**	**.44**	**.44**	**.44**	**.44**	**.43**	.50	.50
pass	.51	.50	.50	.50	.50	.50	.51	**.44**	**.44**
No.	19	20	21	22	23	24	25	26	27
2-x	.50	.50	.51	.50	.50	.50	.50	.50	.50
3-x	.50	.50	.50	.50	.50	.50	.50	.50	.49
pass	**.45**	**.45**	**.44**	**.44**	**.44**	**.44**	**.43**	**.44**	**.44**

$$P_4 = Sh(L_1, \texttt{I}) \otimes Sh(L_2, \texttt{II}) \otimes Sh(L_1, \texttt{III}) \otimes Sh(L_4, \texttt{IV}) \otimes Sh(L_1, \texttt{V}).$$

Now when VSA representations of the patterns are constructed we are ready to construct the classifier in form (1).

5.2 Operating mode

When the classifier is constructed by bundling all patterns bound to their corresponding classes, it is ready for its main operation. The main operation is obviously formulated as testing an input pattern (P) on inclusion in the VSA classifier. The result of the classification is the class to which the input pattern belongs to: $Type = P \otimes CLASSIFIER$.

The result of the above operation is the noisy version of the HD-vector encoding the type, which the pattern belongs to. The remaining operation is therefore performing a Hamming distance test with the clean versions of the type-representing HD-vectors:

$\Delta_H(Type, T_1) = \|Type \otimes T_1\|_1$;
$\Delta_H(Type, T_2) = \|Type \otimes T_2\|_1$;
$\Delta_H(Type, T_N) = \|Type \otimes T_N\|_1$.

Applied to our showcase scenario, the results of the classification tests are shown in Tab. 2. As it is visible from the table the Hamming distance test for each pattern correctly identifies the type, which the pattern belongs to ($\Delta_H << 0.5$). This result confirms our hypothesis for suitability of the VSA for classification of temporal patterns.

6 Discussion

First, we comment on the performance of the VSA-based classification in the time domain. The processing time for classification operation of course depends on the device, which runs the classifier. For classification of one pattern it takes approximately 13 basic

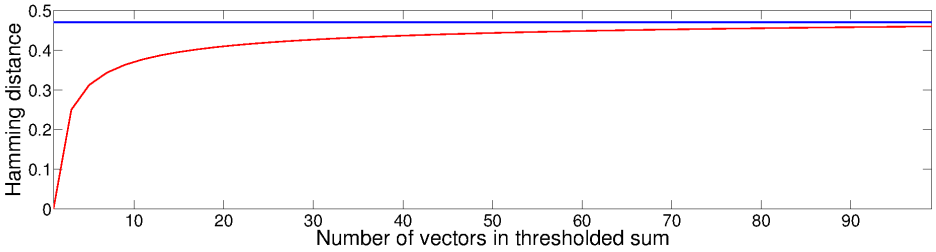


Figure 5: Distance to a random HD-vector included in the thresholded sum operation.

arithmetic operations on HD-vectors. As a test case we implemented the classifier using 1kB HD-vectors on a low-end sensor platform featuring 16-bits microcontroller. In this implementation one classification operation took approximately 130 milliseconds. Obviously, the performance could be boosted dramatically with direct implementation of the repeating operations directly in the hardware.

There is a natural limit on the number of patterns, which can be bundled into one VSA representation while still making it possible to do the decoding operation with required accuracy. This limitation comes from the noise-inducing properties of the summation operation [2]. As it is illustrated in Fig. 5, for 1kB HD-vectors the distance to a random HD-vector included in the thresholded sum operation can deviate to 0.47. This result means that on this dimension a compositional VSA structure can contain around 100 bundled vectors. In the case of our classifier this means that it is able to "store" 100 historical records still maintaining the appropriate accuracy of the classification. Whether this result is positive or negative is difficult to judge at this moment. For the sake of discussion we conjecture that for some applications this number of distinct patterns included in the classifier is sufficient. It is also possible to increase the number of patterns by using for example HD-vectors of higher dimensions (hundreds kilobytes, for example), the effect of this on the time performance and the overall complexity should be additionally studied. We leave the development of this issue for our future work.

7 Conclusion and future work

In this article we presented the system for brain-like classification of temporal patterns based on adopting Vector Symbolic Architectures and Binary Spatter Codes for representing continuous signals. To the best of our knowledge this is the first attempt of this kind. We demonstrated feasibility of using VSA representation at least as a memory efficient storage of temporal patterns. The proposed VSA-based classifier is capable of storing distributed images of up to 100 samples when building it using BSC codewords of size 1kB. We demonstrated that even without the self-learning capabilities, the proposed classifier has real-life application scenarios: We used the classifier to analyse measurements of vehicle passages from the vibration sensor. In some application the proposed

classifier can be implemented in low-end hardware while still demonstrating sufficient performance. The learning capabilities of will further enrich the functionality of the proposed classifier. The development of this topic we leave, however, for our future work.

Acknowledgements

This work is partially supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), institutional grant IG2011-2025.

References

[1] D. G. S. Richard O. Duda, Peter E. Hart, *Pattern Classification*, 2nd ed. John Wiley and sons, LTD, 2000.

[2] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, October 2009.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[4] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 8, pp. 957–961, December 1996.

[5] D. W. Arathorn, "Cortically plausible inverse problem method applied to complex perceptual and planning tasks," in *Proceedings SPIE Defense and Security Symposium*, 2006.

[6] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Transactions on Neural Networks*, vol. 19, pp. 212–229, February 2008.

[7] S. D. Levy and R. Gayler, "Vector symbolic architectures: A new building material for artificial general intelligence," in *Proceedings of the 2008 Conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008, pp. 414–418. [Online]. Available: http://dl.acm.org/citation.cfm?id=1566174.1566215

[8] K. P, K. J, and H. A, "Random indexing of text samples for latent semantic analysis," in *Proc. 22nd annual conference of the Cognitive Science Society*, 2000, p. 1036.

[9] M. Sahlgren, "An introduction to random indexing," in *Methods and Applications of Semantic Indexing Workshop at the 7th TKE Conference*, 2005.

[10] F. Sandin, B. Emruli, and M. Sahlgren, "Incremental dimension reduction of tensors with random index," Cornell University Library, 2011, http://arxiv.org/abs/1103.3585.

[11] S. Levy, S. Bajracharya, and R. Gayler, "Learning behavior hierarchies via high-dimensional sensor projection."

[12] D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, "Dependable mac layer architecture based on holographic data representation using hyper-dimensional binary spatter codes," in *Multiple Access Communications : 5th International Workshop, MACOM 2012*. Springer, 2012, pp. 134–145.

[13] P. Jakimovski, H. R. Schmidtke, S. Sigg, L. W. F. Chaves, and M. Beigl, "Collective communication for dense sensing environments," *Journal of Ambient Intelligence and Smart Environments (JAISE)*, vol. 4, no. 2, pp. 123–134, March 2012.

[14] T. Plate, *Distributed Representations and Nested Compositional Structure*. University of Toronto, PhD Thesis, 1994.

[15] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, 1988.

[16] B. Popovic, "Generalized chirp-like polyphase sequences with optimum correlation properties," *Information Theory, IEEE Transactions on*, vol. 38, no. 4, pp. 1406–1409, 1992.

Paper B

Holographic Graph Neuron: A Bio-Inspired Architecture for Pattern Processing

Authors:
Denis Kleyko, Evgeny Osipov, Alexander Senior, Asad I. Khan, and Y. Ahmet Şekercioğlu

Reformatted version of paper submitted to:
Neural Networks and Learning Systems, IEEE Transactions on.

Holographic Graph Neuron: A Bio-Inspired Architecture for Pattern Processing

Denis Kleyko, Evgeny Osipov,
Alexander Senior, Asad I. Khan, and Y. Ahmet Şekercioğlu

Abstract

In this article, we propose a new approach for implementing Hierarchical Graph Neuron, an architecture for memorizing patterns of generic sensor stimuli, through the use of Vector Symbolic Architectures. The adoption of a vector symbolic representation ensures a single-layer design, while retaining the existing performance characteristics of Hierarchical Graph Neuron. This approach significantly improves the noise resistance of the Hierarchical Graph Neuron architecture, and enables a linear (with respect to the number of stored entries) time search for an arbitrary sub-pattern.

1 Introduction

Graph Neuron (GN) is an approach for memorizing patterns of generic sensor stimuli for later template matching [1, 2, 3]. It is based on the hypothesis that a better associative memory resource can be created by changing the emphasis from high speed sequential CPU processing to parallel network-centric processing. In contrast to contemporary machine learning approaches, GN allows introduction of new patterns in the learning set without the need for retraining. Whilst doing so, it exhibits a high level of scalability i.e. its performance and accuracy do not degrade as the number of stored patterns increases over time.

Vector Symbolic Architectures (VSA) [4] concept is a bio-inspired method of representing concepts and their meanings for modeling cognitive reasoning. It exhibits a set of unique properties which makes it suitable for implementation of artificial general intelligence [5, 6, 7], and so, creation of complex systems for sensing and pattern recognition without reliance on complex computation. In the biological world, extremely successful applications of such approaches can be found. One example is the ordinary house fly: It is capable of conducting very complex maneuvers even though it possesses very little computational capacity[1]. Another interesting biological example is the compound eyes

[1]In [8], a house fly's properties are compared and contrasted with an advanced fighter plane as follows: "Whereas the F-35 Joint Strike Fighter, the most advanced fighter plane in the world, takes a few measurements - airspeed, rate of climb, rotations, and so on and then plugs them into complex equations, which it must solve in real time, the fly relies on many measurements from a variety of sensors but does relatively little computation."

of arthropods. These compound eyes consist of large number of sensors with limited and localized processing capabilities for performing relatively complex sensing tasks [9].

This article presents contributions in two domains: Organization of associative memory, and properties of connectionist distributed representation. In the first area the article introduces a novel bio-inspired architecture, called Holographic Graph Neuron (HoloGN), for one-shot pattern learning, which is built upon Graph Neuron's flexible input encoding abstraction and strong reasoning capabilities of the VSA representation. In the second area, the article extends understanding of the performance proprieties of distributed representation, which opens the way for new applications.

The article is structured as follows: Section 2 places HoloGN in the scope of associative memory research. Section 2 presents an overview of the work related to the matter presented in this article. The background information on the theories, concepts and approaches used in HoloGN is described in Section 4. Sections 5 through 7 present the main contribution of this article, the design of the HoloGN architecture and its performance characteristics. They are followed by our concluding remarks (Section 6).

2 HoloGN in the scope of associative memory research

This section presents a discussion which places HoloGN in the scope of other associative memory approaches. Associative memory (AM) is designed for applications requiring fast pattern matching. Compared to random access memory in modern computing architectures, where the goal is to retrieve the content of a certain place of the memory by supplying the address of this place, the goal of the AM is different. The cue to the memory is a pattern in a generic sense[2], which is stored in memory either entirely or in parts. The task is to search the entire memory for the best matching entry. On the very high level, the taxonomy of AM includes *localist* and *distributed* approaches.

The localist-based AM models [10] rely on the so called "grandmother cell" hypothesis, where each part of a pattern considered by the model are represented with a single neuron. In studies of perception, such models, in spite of known criticism, could be plausible in certain biological systems in which a wide range of neurons prefer specific stimuli, e.g. in [11] it was shown that a population of a few tens of neurons in a monkey's brain were selectively responsive to different features of their body.

The second broad class of AM models, which are based on distributed representations, oppose the localist-AMs by suggesting that for implementing association-based operations on structured information it is implausible to have dedicated neurons for each element of the structure. In *distributed representation theory* a specific stimulus is coded by a unique pattern of activity over a group of neurons.

The work presented in this article has its roots and inspiration in two specific models of the distributed associative memory: Sparse Distributed Memory (SDM) [12] and Hi-

[2]The term "pattern" is used here to represent a set of (heterogeneous) values or concepts that repeat over time to form an experience of an artificial system.

erarchical Graph Neuron (HGN) [2]. SDM is a mathematical model of human long-term memory, which is used for storing and retrieving large amounts (in the order of 2^{1000} bits) of information. Its two main principles are aggregation of similar stimuli based on statistical similarity of their encoded representations and reducing the overlap between the representations by storing them sparsely over a huge memory space. SDM was extensively applied to pattern recognition [13], as the model for implementing AM in the context of cognitive computing architectures [14] as well as its demonstrated suitability for approximating Bayesian inference [15]. One of the *unsolved* challenges attributed to SDM is the so-called *encoding problem*, i.e. the problem of how to encode stimuli into the distributed representation [16].

HGN can be classified as a distributed AM in the sense that it models a stimulus as a graph of activities of multiple elementary neurons. Note that in HGN the definition of a neuron is simplified: it is modeled as an array of possible values taken from a finite alphabet of discrete values. Without discussing the biological plausibility of this model, it is applicable in many practical real-world scenarios. For example all current sensing devices are characterized by having a finite operating range, and quantization techniques (i.e. representing the level of the sampled continuous signal in finite number of levels) suggest this model is feasible. Practically this model of a neuron and an interconnected network of them enables lightweight implementation of AM-based sensor networks using low-end and power-constrained computing devices.

The work presented in this article provides a step towards addressing the encoding problem of SDM. Specifically, it demonstrates that a simple (in the computational sense) but usable model of a neuron from the HGN approach leads to practical implementation of an associative memory using high-dimensional representations. By doing so, we eliminate the need for maintaining a graph hierarchy by applying the high-dimensional representations and mathematical apparatus of SDM for modeling congregation of memories based on statistical similarities between the encoded concepts. As a result, the encoding of an acquired heterogeneous sensory stimulus not only preserves the properties of the original model, but also paves the way to an entirely new class of applications such as in [17, 18].

3 Related Work

Associative memory is a sub-domain of artificial neural networks, which utilises the benefits of content-addressable memory (CAM) [19] in microcomputers. The AM concept was originally developed in an effort to utilise the power and speed of existing computer systems for solving large-scale and computationally intensive problems by simulating biological neurosystems.

The Hierarchical Graph Neuron (HGN) approach [2] is a type of associative memory which signifies the hierarchical structure in its implementation. Hierarchical structures in associative memory models are of interest as these have been shown to improve the rate of recall in pattern recognition applications. The distributed HGN scheme also allows for better control of the network resources. This scheme compares well with contemporary

approaches such as Self-Organizing Map and Support Vector Machine in terms of speed and accuracy.

Vector Symbolic Architectures (VSA) concept were introduced by Levy and Gayler [5] as a class of connectionist models that use hyper-dimensional vectors (i.e. vectors of several thousand elements) to encode structured information as a *distributed* or *holographic* representation. In this technique structured data is represented by performing basic arithmetic operations on field-value tuples. Distributed representations of data structures are an approach actively used in the area of cognitive computing for representing and reasoning upon semantically bound information [4, 20]. The cognitive capabilities achievable using VSAs have been demonstrated by creating systems capable of solving Raven's progressive matrices [21, 22] and via imitation of concept learning in honey bees [23, 24].

In [25] a VSA-based knowledge-representation architecture is proposed for learning arbitrarily complex, hierarchical, symbolic relationships (patterns) between sensors and actuators in robotics. Recently the theory of hyper-dimensional computing, and VSA in particular, were adopted for implementing novel communication protocols and architectures for collective communications in machine-to-machine communication scenarios [26, 27]. The first work demonstrates unique reliability and timing properties essential in the context of industrial machine-to-machine communications. The latter work shows the feasibility of implementing collective communications using current radio technology. This article presents an algorithmic ground for further design of the distributed HoloGN on top of the architecture presented in [26].

4 Overview of essential concepts and theories

4.1 Hierarchical Graph Neuron

Figure 1 illustrates the Hierarchical Graph Neuron (HGN) approach. Consider only the bottom layer of the construction without the hierarchy of upper nodes; this bottom level is the original *flat* network of Graph Neurons [2]. Each GN is a model for a set of generic sensory values (e.g. the value of a pixel or a real-value of sensory data). When seen as a network, graph neurons can be modeled by an array where columns are individual GNs and rows are possible symbols, which a neuron can recognize, e.g. an integer between 0 and 100. For example, if there are only two possible symbols, say "X" and "Y" in the alphabet of a pattern, then only two rows are needed to represent those symbols. The number of columns[3] in the GN array determines the size of patterns which it can analyse.

An input pattern is defined as a stimulus produced within the network. In Figure 1, each GN can analyse a symbol ("X" or "Y") of a pattern consisting of five elements. In each GN (column) only the element with the matching value (a row ID) would respond. For example, if the pattern is "YXYYX", then in the second column the "X" element will

[3]In this article the words "column" and "GN" are used interchangeably and refer to a single Graph Neuron. The term "GN array" refers to several GNs used to recognize a pattern of several elements, where one neuron is used to recognize one element of the pattern.

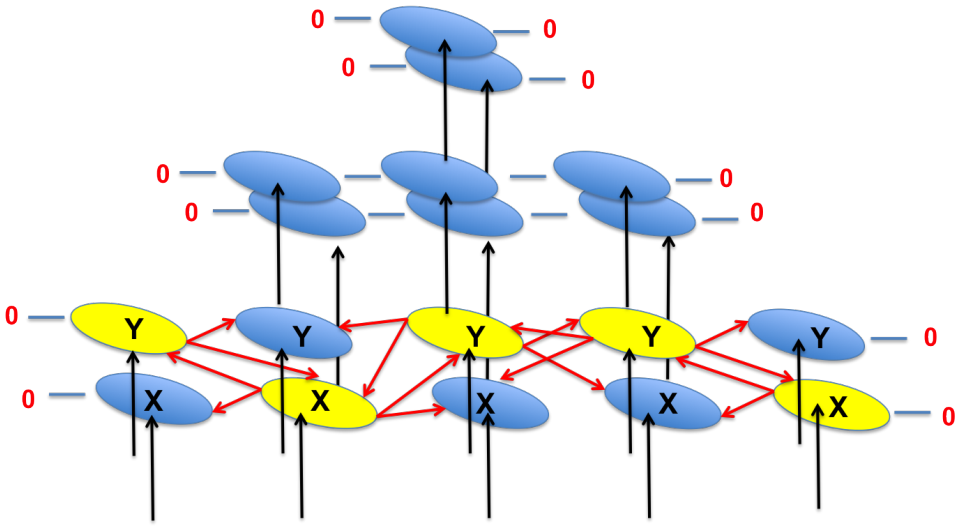


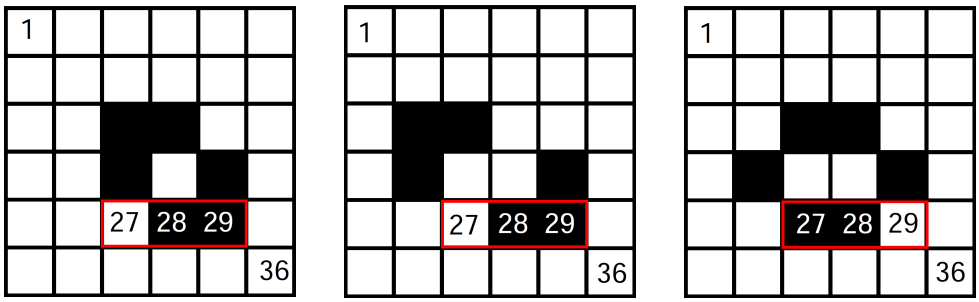
Figure 1: Hierarchical GNs of a five element pattern of two symbols. The short arrows in the figure show how GNs communicate indices of the activated GN-elements with their neighbours creating logical connectivity. Null GN numbers are assigned at the start and at the end of the array for maintaining a consistent reporting scheme where every activated GN reports to both the adjacent GNs.

be activated as response to this input stimulus. If a particular element in the GN-column is activated it sends a report to all adjacent GNs. The report contains the activated GN's element ID (the row index). Otherwise, it simply ignores the stimulus and returns to the idle state.

During the next phase, all GNs communicate the indices of the activated elements with the adjacent columns at their level, and additionally communicate the stored bias information to the layer above. The procedure continues in an ascending manner until the collective bias information reaches the top of the hierarchy. The higher level GNs can thus provide a more authoritative assessment of the input pattern. The accuracy of HGN was demonstrated to be comparable to the accuracy of Neural Network with back-propagation [2].

4.2 An example of HGN operation

In order to give a better intuition behind the encoding procedure of the original Hierarchical Graph Neuron, consider a task of memorizing primitive pixel patterns as illustrated in Figure 2. The patterns are 6x6 black and white pixel images. Suppose that the pixels are enumerated from 1 to 36 starting from the upper left corner. To facilitate the presentation each pattern is given a name of "still life" figures in the Game of Life [28].



(a) The 'ship'. (b) The 'aircraft carrier'. (c) The 'beehive'.

Figure 2: Three examples of pixel images for HGN memorization. These patterns are named after 'still life' patterns in the Game of Life. The highlighted cells have ordered indices 27, 28, and 29 and are referred to in the text.

That is pattern (a) depicts the 'ship', pattern (b) depicts the 'aircraft carrier', and (c) depicts the 'beehive'. The patterns will be subsequently presented to the hierarchical graph neuron in the order 'ship', 'carrier' and 'beehive'.

The bottom layer of the HGN consists of an array of 36 GNs where each GN_i is dedicated to recognizing the state (on/off) of the corresponding pixel. The subscript i corresponds to the number of the pixel, which this GN is assigned to. GN_i is a column with two elements $GN_i[0]$ ='white' and $GN_i[1]$ ='black'. In what follows the operation of the three GNs highlighted by the red rectangle in the figure is considered, i.e GN_{27}, GN_{28} and GN_{29}.

The first presented pattern ('ship') results in activations $GN_{27}[0]$, $GN_{28}[1]$ and $GN_{29}[1]$, i.e. white-black-black. The activated indices will be communicated by each GN to their immediate neighbors. Each GN then notes down which neighbors were activated in a table of information called the *bias array*. The bias array essentially links a certain activation of neighbors to an integer index of the record. In this example, the middle 'black' element ($GN_{28}[1]$) will associate the activation of its white neighbor on its left and its black neighbor on its right with an index of 0, for example. It will then communicate this index to the element directly above it in the hierarchy of layers (as seen in Figure 1). This (upper) element will in turn broadcast its activation to its neighbors and receive similar messages, and create a new entry in its own bias array. This process will continue until it reaches the uppermost layer in the GN hierarchy, consisting of a single column of two elements.

Now consider the 'aircraft carrier' and 'beehive' images; as the highlighted pixels in the 'aircraft carrier' (Figure 2b) are the same between the two images, when the three elements broadcast their activation and consult their bias arrays, they will find that they encountered the same activation before, and hence will emit the same index to the higher layer. However, in the 'beehive' image (Figure 2c) the activation will now be 'black-black-white' instead of 'white-black-black'. This means that $GN_{27}[1]$ and $GN_{29}[0]$ will be activated for the first time. They will form entries in their (empty) bias arrays in a

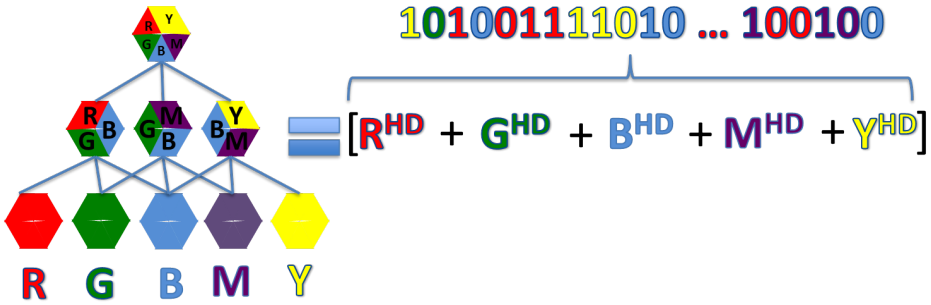


Figure 3: A High level illustration of the proposed solution: representation of the Hierarchical Graph Neuron using Vector Symbolic Architecture; where the letters depict: R - red; G - green; B - blue; M - magenta; Y - yellow.

similar fashion as before and transmit the index (0) to the assigned node of the upper hirarchical level. Meanwhile, the black element in the middle ($GN_{28}[1]$) will activate as before, but as its activated neighbors are different, it will form a new entry in its bias array with an index of 1, and transmit this information to the next upper layer. In this way, the differences and similarities between patterns are stored in a distributed manner throughout the hierarchy of GNs.

4.3 Motivation for Holographic Graph Neuron

An important issue in hierarchical models is the overhead of resource requirements, specifically with regards to the number of processing elements required. For example if the HGN for recognition of patterns of 5 elements (as in Figure 1) is to be implemented in a wireless sensor network, where a sensor node is a single GN, then 9 sensor nodes are required, but only 5 are actually used to observe patterns (bottom layer). This paper proposes a holographic approach, which (1) borrows the abstraction of the Graph Neuron; and (2) enables a flat GN array to operate with higher level of accuracy and comparable recall time than that of HGN, without the need for a complex topology and additional nodes. The high-level logic of the proposed solution is presented in Figure 3. Figure 3 illustrates the ideas of both approaches. On the left HGN creates the representation of a pattern through communication indices of the activated GN-elements between neighbors. Thus, on the highest level HGN forms representation of the whole pattern (illustrated with mixed colors in the figure). In contrast, on the right HoloGN achieves similar result by encoding each GN element and the compination of their particular activations into a distributed representation. The fundamentals of the algebra of distributed representations are presented in the next subsection.

4.4 Fundamentals of Vector Symbolic Architecture and Binary Spatter Codes

As mentioned previously, VSA is an approach for encoding and operations on distributed representation of information, and has previously been used mainly in the area of cognitive computing for representing and reasoning upon semantically bound information [4, 29].

The fundamental difference between distributed and localist representations of data is as follows: in traditional (localist) computing architectures each bit and its position within a structure of bits are significant (for example, a field in a database has a predefined offset amongst other fields and a symbolic value has unique representation in ASCII codes); whereas in a distributed representation all entities are represented by vectors of very high dimension. For the remainder of the article the term HD-vector is used when referring to such codes. In particular, the Binary Spatter Code (BSC) variety of HD-vectors is utilized. *High dimensionality* refers to that fact that in HD-vectors, several thousand positions (of binary numbers) are used for representing a single entity; [4] proposes the use of vectors of 10000 binary elements. Such entities have the following useful properties:

Randomness

Randomness means that the values at each position of an HD-vector are independent of each other, and "0" and "1" components are equally probable. In very high dimensions, the distances from any arbitrary chosen HD-vector to more than 99.99 % of all other vectors in the representation space are concentrated around 0.5 normalized Hamming distance. Interested readers are referred to [4] and [12] for comprehensive analysis of probabilistic properties of the hyperdimensional representation space.

Denote the density of a randomly generated HD-vector (i.e. the number of ones in a HD-vector) as k. The probability of picking a random vector of length d with density k, where the probability of 1's appearance equals p is described by the binomial distribution:

$$\Pr(k, d, p) = \binom{d}{k} p^k (1-p)^{d-k}. \tag{1}$$

When d is in the range of several thousand binary elements, the calculation of the binomial coefficient requires sophisticated computations. Therefore, approximations of binomial distribution are used for large values of d. It can be well approximated via the normal approximation, the de Moivre-Laplace theorem or Poisson distribution (for sparse vectors). The calculations in this paper use the de Moivre-Laplace theorem.

Similarity Metric

The similarity between two binary representation is characterized by normalized Hamming distance, which (for two vectors) measures the number of positions in which they

differ:

$$\Delta_H(A,B) = \frac{1}{d}\|A \otimes B\|_1 = \frac{1}{d}\sum_{i=0}^{d-1} a_i \otimes b_i, \tag{2}$$

where a_i, b_i are bits on positions i in vectors A and B of dimension d, and where $\otimes$ denotes the bit-wise XOR operation.

Generation of HD-vectors

Several random binary vectors with the above properties can be generated from one such vector via the cyclic shift operation. Using this operation a sequence of K vectors, which are pseudo-orthogonal to a given initial random HD-vector A (i.e. normalized Hamming distance between them approximately equals 0.5) are obtained by cyclically shifting A by i positions, where $1 \leq i \leq K < d$. Further in the article this operation is denoted as $\mathrm{Sh}(A,i)$. The cyclic shift operation has the following properties:

- it is invertible, i.e. if $B = \mathrm{Sh}(A,i)$ then $A = \mathrm{Sh}(B,-i)$;
- it is associative in the sense that $\mathrm{Sh}(B,i+j) = \mathrm{Sh}(\mathrm{Sh}(B,i),j) = \mathrm{Sh}(\mathrm{Sh}(B,j),i)$;
- it preserves Hamming weight of the result: $\|B\|_1 = \|\mathrm{Sh}(B,i)\|_1$; and
- the result is dissimilar to the vector being shifted: $\frac{1}{d}\|B \otimes \mathrm{Sh}(B,i)\|_1 \approx 0.5$.

Note that the cyclic shift is a special case of the permutation operation [4]. In the context of VSA, permutations were previously used to encode sequences of semantically bound elements.

Bundling of Vectors

Joining several entities into one structure is done with the bundling operation; it is implemented by a thresholded sum of the HD-vectors representing the entities. A bit-wise thresholded sum of n vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. In the case of an even number in sum ties are broken at random. This is equivalent to adding an extra random HD-vector [4]. Furthermore the terms "thresholded sum" and "majority sum" are used interchangeably and denoted as $[A + B + C]$. The relevant properties of the majority sum are:

- for any number of operands the result is a vector, with the number of '1' components is approximately equal to the number of '0' components;
- the result is similar to all vectors included in the sum;
- the more HD-vectors that are involved in a majority operation, the closer the normalized Hamming distance between the resultant vector and any HD-vector component is to 0.5; and

- if several copies of any vector are included into a majority sum, the resultant vector is closer to the dominating vector than to other components.

The algebra on VSA includes other operations e.g., binding, permutation [4]. Since they are not used in this article, their definitions and properties are omitted.

5 Holographic Graph Neuron

This section presents one of the main contributions of this paper: the adoption of the VSA data representation for implementation of the HGN approach.

The commented Matlab implementation of HoloGN and simulation scenarios used in this article to produce Figures 8–10) are available online. A `git` user can obtain the code by using the command `git clone https://github.com/eaoltu/hologn.git`. The readers who are not familiar with `git` can download the code from `https://sites.google.com/site/evgenyosipov/professional/research-projects/hologn`. The included Readme.txt file contains the details on how to work with the code. For the sake of saving space, the code snippets are not included in the article, the readers are assisted with the references to the particular functions in the implementation.

5.1 Encoding

In the case of HoloGN, all its elements i.e. symbols of individual neurons (e.g. possible values of image pixel) are indexed uniquely and the index of a particular element is derived as a function of the GN's ID. Let IV_j be an *initialization high-dimensional vector* for GN j. The initialized vectors for different GNs are chosen to be mutually orthogonal. Then the HD-index of element i in GN j is computed as $E^{HD}_{(j,i)} = \mathrm{Sh}(IV_j, i)$, where Sh() is a cyclic shift operation resulting in the generation of a vector orthogonal to IV_j HD-vector [30]. [4]

5.2 Construction of VSA-representation of Activated GNs

Let n be the number of individual Graph Neurons. When a GN array (n GNs) observes a pattern, the activated elements communicate their HD-represented indices to all other GNs; the holographic representation of the activated elements is then

$$\mathrm{HGN} = \left[\sum_{j=1}^{n} (E_j^{\mathrm{HD}}) \right], \tag{3}$$

where E_j^{HD} is the HD-index of the activated element in $\mathtt{GN}_j$, and the addition operation is the bundling operation, or thresholded sum as described in 4.4. [5] As discussed previously,

[4] In the implementation the encoding is done in the *hologn_encoder* function.

[5] In the HoloGN code the majority sum is implemented in the *majority_sum* function. This function is then used to construct the HGN as in (3).

in the resultant HD-vector the normalized Hamming distances between each component and the composition vector strictly less are than 0.5. This property will be utilized later when constructing data structures for recall of patterns in HoloGN.

5.3 Data Structures for Storing and Retrieving Holographic Representations in HGN Elements

HoloGN will store the holographic representation of the entire pattern (3) observed across all GNs. A possible architecture is for all memorized patterns to be collected and stored centrally at a processing node, where the role of processing node can be assigned to one of GNs or to some other external device.

Depending on the particular application of HoloGN, the memorized patterns could be stored either in an unsorted list or in bundles. The first mode of storing HoloGN patterns corresponds to the case where the structure of the observed patterns is unknown; the latter mode is used in the case of a *supervised learning*. The next section describes different HoloGN usage and recall strategies.

6 HoloGN Recall Strategies

This section introduces and evaluates the performance of two major recall modes: the one-shot case and the case of supervised learning.

6.1 Time-efficient ξ-accurate Recall in an Unsorted HoloGN Storage

The common step in both recall modes is the procedure for the time-efficient search over an (unsorted) list of HoloGN records. Recall that all manipulations with VSA-encoded entities are done using simple bit-wise arithmetic operations as well, and calculations to obtain the Hamming distance between entities. However, it is assumed for this article that there is no particular optimized implementation of VSA's bit-wise operations; this is because such operations are tailored to the architectures of specific microprocessors, which operate with words of substantially lower dimensionalities (typically 32 or 64 bits). Therefore, adopting these methods for implementing the bit-wise operations on words of thousands of bits would be cumbersome. Instead, an easily analyzable computational model is adopted in this article, which could also be adapted to an implementation on specialized computing architectures.

In what follows, each HoloGN pattern $\mathbf{h}_i$ is modelled as a row vector of d elements. The list of stored HoloGN patterns is therefore modelled as an $l \times d$ matrix, where l is the number of the learned (stored) HoloGN patterns. Denote this matrix as $\mathbf{H}$. The task of recalling a pattern with a target recall threshold of ξ ($\xi < 0.5$) is formulated as finding the rows $\mathbf{h}_i$ in $\mathbf{H}$ with normalized Hamming distances to the query pattern $\mathbf{h}_q$ less than or equal to ξ.

The conventional way of computing normalized Hamming distance between vectors would be to perform the following sequence of computations for each row in $\mathbf{H}$:

1. perform an element-wise XOR with the vector query;
2. sum up all elements in the intermediate result; and then
3. divide the result by the dimensionality of the vectors.

The performance of the two implementations of this method using Matlab's `repmat` and `bsxfun` functions are shown by the top two curves in Figure 4; `bsxfun` applies element-by-element binary operation to two arrays with singleton expansion enabled. The curves demonstrate linear but rapid increase in the recall time with the increase in the number of the stored patterns. The lowest curve in the figure shows the performance of matrix-vector multiplication of the same size, which is chosen as the reference case. The results were obtained on a Intel Core i7-3520M 2.9 GHz machine with Windows 7 operating system using one processor.

Binary Spatter Codes as Complex Numbers

In order to improve the efficiency of calculating Hamming distances over a vast number of HoloGN patterns, it is proposed to represent HoloGN patterns using complex numbers, where a binary 0 would be represented by $\sqrt{-1}$ (i.e. the complex number); and a binary 1 would remain 1. The intuition behind this transformation is simple: multiplication of bits in the same position should produce three outcomes: $-1 = j \times j$, $1 = 1 \times 1$ and $j = 1 \times j$. That is, the multiplication of two similar bits would produce a real number and the multiplication of two different bits produces a imaginary number. In this way the sum of the imaginary parts over all positions in the resulting vector, divided by dimensionality d, will correspond to the normalized Hamming distance between the two vectors. Thus, the suggested method allows us to implement the calculation of Hamming distance through the standard method of matrix-vector multiplication, as illustrated below:

$$\mathbf{H} \times \mathbf{h}_q = \begin{pmatrix} \sqrt{-1} & 1 & \cdots & \sqrt{-1} \\ 1 & 1 & \cdots & \sqrt{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{-1} & \sqrt{-1} & \cdots & 1 \end{pmatrix} \times \begin{pmatrix} \sqrt{-1} \\ 1 \\ \vdots \\ \sqrt{-1} \end{pmatrix}$$

$$= \begin{pmatrix} 254 + 1633j \\ 617 + 3824j \\ \vdots \\ 548 + 4952j \end{pmatrix}. \tag{4}$$

The performance of the proposed method is illustrated by the dashed curve in Figure 4. It demonstrates that the calculation of the Hamming distance is only two times slower

than the usual matrix-vector multiplication. Specifically, to Hamming distance from the target vector to each of 20 000 stored patterns takes approximately 200 ms on the test machine. Further optimization of the matrix multiplication and execution on parallel architectures hint on realistic bounds on the recall time over extremely large numbers of patterns. [6]

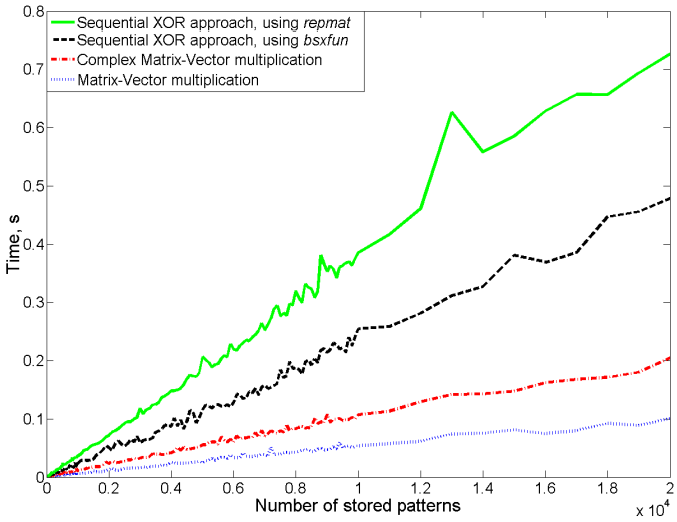


Figure 4: A comparison of the different approaches to recalling patterns, showing the time taken to calculate the Hamming distance against the number of previously presented patterns.

6.2 Case-study 1: Best-match Probing Under One-shot Learning

The first usage of HoloGN is probing the existence of the target query pattern amongst the previously memorized patterns. The perfect match in this case would be indicated by a normalized Hamming distance of zero. The deviation from zero, therefore, reflects the degree of proximity of the query to one or several stored HoloGN patterns. [7] In the following example the accuracy of the HoloGN recall was compared to the performance of the original HGN approach. For the sake of fair comparison the 7 by 5 pixels letters

[6] In the HoloGN implementation the transformation of the array of binary values into the array of complex values is done by the *bin2com* function.

[7] The recall of the closest memorized pattern for a given representation of the target query is implemented in the *item_memory_c* function.

of the Roman alphabet (as in [2]) were used.

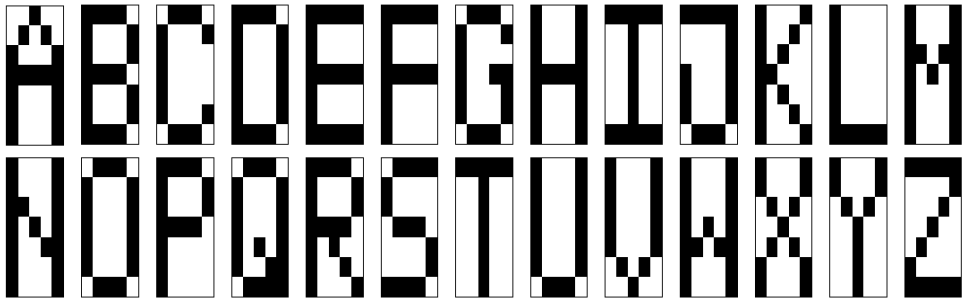


Figure 5: List of letters images for comparison.

In the memorization phase a set of noise-free images of letters illustrated in Figure 5 was presented to both architectures. In the recall phase images of the same letters distorted with different levels of random distortions (between 1 bit corresponding to a distortion of 2.9% of the pattern's size and 5 bits equivalent to 14.3% distortion) were presented to the architectures for the recall. An example of a noisy input is presented in Figure 6. In the case of HoloGN the pattern with the lowest normalized Hamming distance to the presented distorted pattern was returned as the output.

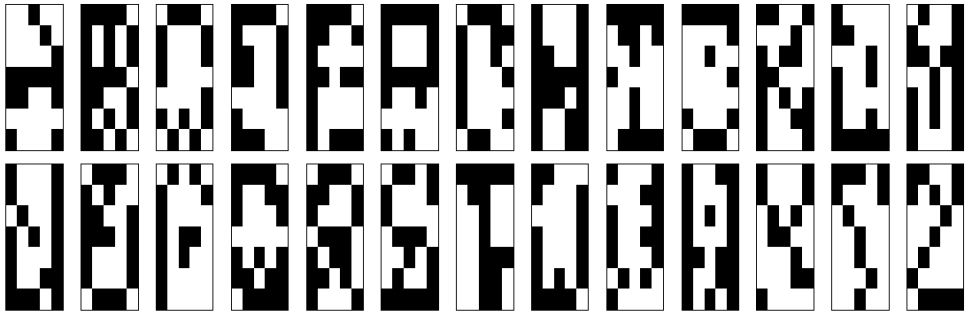


Figure 6: An example of presented for recall images distorted by 5 bits (14.3%).

Figure 8 presents the results of the accuracy comparison between the recall results for the HoloGN approach and the reference HGN architecture. To obtain the results 1000 distorted images of each letter for every level of distortion were presented for recall. The charts show the percentage of the correct recall output. The analysis shows that the performance of the HoloGN-based associative memory at least matches that of the original

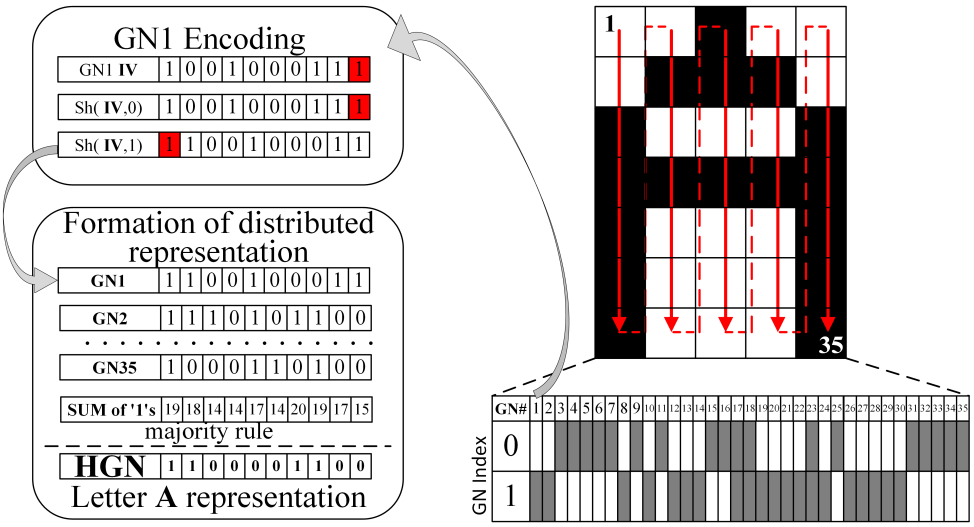


Figure 7: Example of HoloGN encoding for letter "A". For simplicity of presentation the encoding operation is exemplified on 10-dimensional vectors. In the simulations HD-vectors with 10'000 elements were used.

approach. Note, however, that in certain characters the recall is rather inferior to other letters. For example, the accuracy of the character "O" recognition is persistently lower than other letters. This is due its similarity to several other characters. In particular, when recalling "O" distorted by 5 bits HoloGN recall scoring is "C" - 5.2%, "D" - 11.4%, "G" - 11.9%, "Q" - 5.1%. In the performed simulations the average accuracy of HoloGN when recalling patterns is 51.7% higher than the accuracy of HGN.

Example: encoding and recall of letters using HoloGN

Consider 5x7 black and white pixel images of latin letters illustrated in Figure 5. The encoding process is exemplified in Figure 7 and includes the following steps:

- Initialization of HoloGN:
  - Set the dimensionality of the HD-vectors[8]. In this article 10 000 bits were used for the simulations;
  - Set the number of GNs[9]. The image of a letter consists of 35 pixels. Every pixel is assigned one GN, that is 35 GNs are initialized in the simulations;
  - Generate initialization high-dimensional vector IV for each GN[10].

[8]Line 28 in the *hologn_encoder* function.
[9]Line 31 in the *hologn_encoder* function.
[10]Line 36 in the *hologn_encoder* function

- Present a letter-image to the initialized HoloGN[11]. Images for all 26 letters are stored in Letters.mat;
- For each GN (pixel) shift cyclicly this GN's IV to the value of the pixel ('0' for white and '1' for black)[12];
- Form the distributed representation of the letter (see Section 5.2) using shifted IV vectors[13];
- Store the letter's representation in the list of memorized patters[14].

The encoding process is repeated for all 26 letters. Thus at the end of the letters' encoding procedure 26 HD-vectors are created, which are stored in the list of memorized patterns.

The recall phase for a randomly distorted letter is done as follows:

- The pixels of the chosen letter are randomly distorted according to the specified distortion level[15];
- The distributed representation of the distorted pattern is formed as described above using the *letters_encoding* function;
- The processing unit calculates Hamming distances from the representation of the distorted letter to all of the 26 representations stored in the list of memorized patters[16];
- The stored pattern with the minimal Hamming distance to the distorted one is recalled by the HoloGN as the desired letter[17].

Note that the whole simulation scenario for the case-study 1 is available in the file *Scenario_recall_pattens_with_distortions*.

6.3 Case-study 2: HoloGN Recall Under Supervised Learning

The analysis presented above is a very positive result for the proposed bio-inspired associative memory based pattern processing architecture, since the accuracy of the original HGN approach was demonstrated to be as accurate as artificial neural networks with back-propagation [2]. While establishing formal relationships to the framework of artificial neural networks is outside the scope of this work, this section presents the results of the pattern recognition accuracy of the HoloGN architecture under supervised learning[18].

[11]Line 45 in the *letters_encoding* function.
[12]Lines 43–45 in the *letters_encoding* function.
[13]Line 48 in the *letters_encoding* function.
[14]Line 45 in the *letters_encoding* function.
[15]Line 66 in the *Scenario_recall_pattens_with_distortions* scenario.
[16]Line 28 in the *item_memory_c* function.
[17]Line 31 in the *item_memory_c* function.
[18]By supervised learning, we mean labeling distorted patterns by bundling them with the distributed representation of the correct character during the training phase.

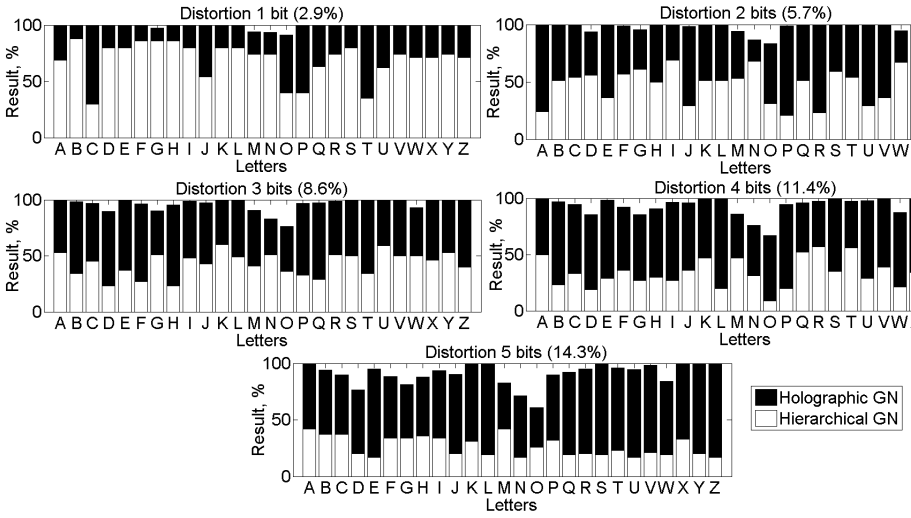


Figure 8: Results from testing black and white images of letters using recall patterns with distortions ranging from 1 bit (2.9%) to 5 bits (14.3%).

In this case the HoloGN is presented with a series of randomly distorted patterns for each letter with different level of distortion (between 1 and 15 bits) as exemplified in Figure 6. In the experiments up to 500 patterns for each letter and every level of distortion were presented for memorizing. For the particular level of distortion i all y presented patterns of the particular letter L_i were bundled to a single HoloGN representation as

$$\mathbf{h}(L) = \left[\sum_{i=1}^{y} (\mathrm{HGN}(L_i)) \right]. \tag{5}$$

Thus by the end of the learning phase the HoloGN list will contain 26 high dimensional bundles, each jointly representing all (presented) distorted variants of the particular letter. In the recall phase for each distortion level HoloGN was presented with 500 new distorted patterns of each letter. The accuracy of the recall was measured as the percentage of the correctly recognized letters averaged over the alphabet.

Figure 9 illustrates the obtained results: 90% accurate recall was observed when learning symbols distorted by up to 5 bits (14.3%). While the accuracy predictably decreases rapidly with the increase of distortion in the presented patterns, a reasonable 80% recall accuracy was observed for learning sets with 7 bits distortion (20%).

Figure 10 illustrates the convergence of the HoloGN recall accuracy with the number of presented noisy samples for the case of 5 bits distortion (14.3%). For larger learning sets the average accuracy in Figure 10 is approaching the average value in Figure 9 for 5 bits distortion. This is a definitely positive result for the presented architecture, which illustrates the suitability of the HoloGN in applications requiring supervised learning.

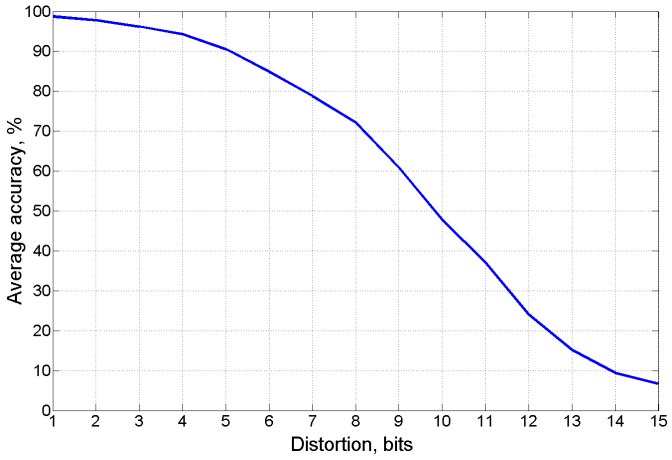


Figure 9: Average accuracy of HoloGN recall under supervised learning memorization as a function of the distortion level.

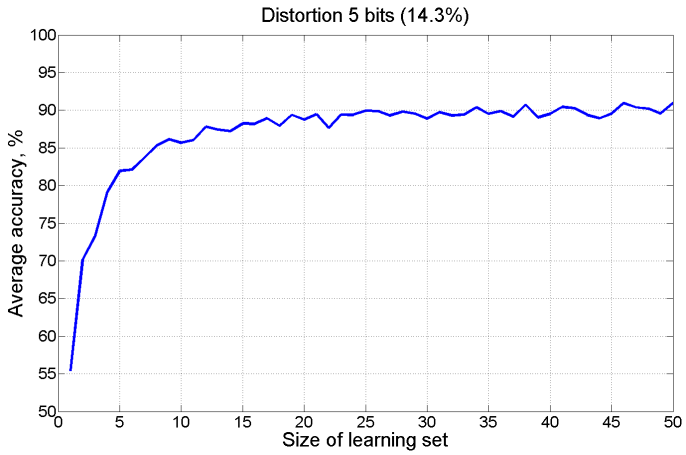


Figure 10: Accuracy of HoloGN under supervised learning memorization as a function of the number of presented examples for a given level of distortion.

7 Pattern Decoding and Subpattern-based Analysis

There is a class of pattern recognition applications which requires an understanding of the details of the recall results. For example, when a recall returns several possible patterns of given recall accuracy, the task would be to understand the overlapping elements. This

section considers two aspects of this task: a robust decoding of elementary components out of a distributed VSA representation; and a quantitative metric of the similarity via direct comparison of distributed representations, without the need for decoding those representations.

The VSA approach of representing data structures by definition makes decoding of the individual components a tedious task, requiring a brute force test on the inclusion of all possible high-dimensional codewords for each GN. The majority sum - which is used for creating HoloGN representations of the observed patterns - imposes a limit on the number high-dimensional codeword operands, above which a robust decoding of the individual operands is impossible.

7.1 Preliminaries

Denote the density of a randomly generated HD-vector (i.e. the number of *ones* in a HD-vector) as k. The probability of picking a random vector of length d with density k, where the probability of 1's appearance, defined as p, is described by (1). The mean density of a random vector is equal to $d \cdot p$. Note that in reality the density of randomly generated HD-vectors will obviously deviate from the mean value. However, according to (1) the density k approaches the mean value with the increase of dimensionality d. In other words, the probability of generating HD-vector with $k \gg d \cdot p$ or $k \ll d \cdot p$ decreases with the increase of dimensionality d.

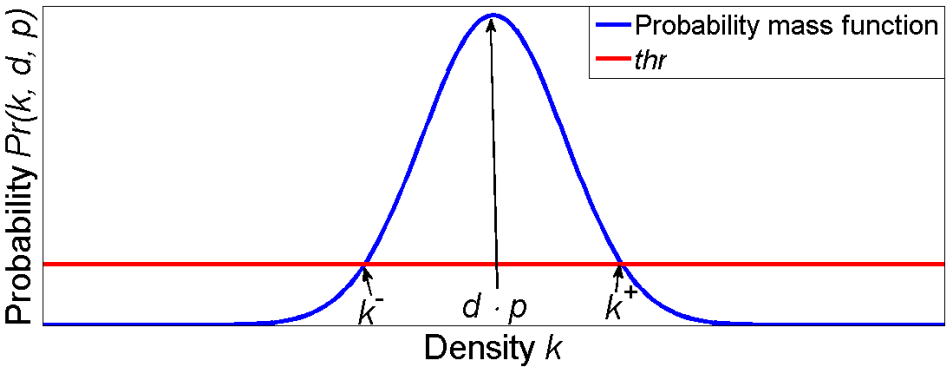


Figure 11: Binomial distribution and its parameters describing HD-vectors.

Define thr as the threshold probability of generating a vector with a certain deviation of density being negligibly small. Let k^- and k^+ characterize the lower and the upper bounds of the interval of possible densities. This is illustrated in Figure 11. The bounds for a given d, p and thr are calculated using (1). The bounds are calculated according to (6) and (7). The value of thr is chosen to be small (10^{-6}).

$$k^{-}(d,p,\mathrm{thr}) = \max_{k}(\Pr(k,d,p) <= \mathrm{thr} | k < (d \cdot p)) \tag{6}$$

$$k^{+}(d,p,\mathrm{thr}) = \min_{k}(\Pr(k,d,p) <= \mathrm{thr} | k > (d \cdot p)) \tag{7}$$

7.2 Capacity of HoloGN Representations

Suppose there exists an item memory [4] containing HD-vectors representing atomic concepts[19]. Recall that when several HD-vectors are bundled by the majority sum the noise of flipped bits increases with the number of components. For a given dimensionality d there is a limit on the number of bundled HD-vectors beyond which the resulting HD-vector becomes orthogonal to every component vector; hence the A *capacity* of the resulting vector is defined as the maximal number of mutually orthogonal HD-vectors which can be robustly decoded from their majority sum composition by probing the item memory. Note, however, that component vectors in fact never become truly orthogonal to the resulting HD-vector, although the component vectors can no longer be reliably extracted from the resulting HD-vector.

In order to characterize the capacity of the composition vector for a given dimensionality, one needs to characterize the level of noise p_n introduced by the bundling operation. This is calculated as in [16] by

$$p_n(n) = \frac{1}{2^n}\left[\frac{1}{2} - \binom{n-1}{0.5 \cdot (n-1)}\right] \tag{8}$$

where n is a number of atomic vectors in the resulting majority sum vector.

Consider an arbitrary HD-vector $\mathbf{A}$ to be decoded from a majority sum composition. Let $\mathbf{N}$ be a vector of noise imposed by the majority sum operation. Since the components are mutually orthogonal, the density of ones in the noise vector is also described the binomial distribution $\Pr(k,d,p_n)$. As each new vector is added the level of noise increases, hence the mean of the noise vector density will approach 0.5 as illustrated in Figure 13. Due to the properties of high dimensional space, vector $\mathbf{A}$ will be undecodable when the upper bound $k^{+}(d,p_n,\mathrm{thr})$ of the density of noise vector $\mathbf{N}$ approaches the lower bound $k^{-}(d,0.5,\mathrm{thr})$. That is, the noisy version of $\mathbf{A}$ becomes orthogonal to its clean version. This logic is illustrated in Figure 12, where the resulting majority sum vector is orthogonal to all components. Thus the capacity of the distributed representation with dimensionality d is computed by

$$\begin{aligned} &\mathrm{Capacity}(d,\mathrm{thr}) = \\ &= \max_{n}(k^{+}(d,p_n(n),\mathrm{thr}) \leq k^{-}(d,0.5,\mathrm{thr})) \end{aligned} \tag{9}$$

Figure 13 presents the capacity of HD-vector of different dimensionalities calculated for threshold probability $\mathrm{thr} = 10^{-6}$. Specifically for $d = 10000$ bits the capacity of the robustly decodable VSA is 89 vectors. Similar analysis of capacity of HD-vectors

[19] In the case of HoloGN an atomic concept is the code for the particular HoloGN element.

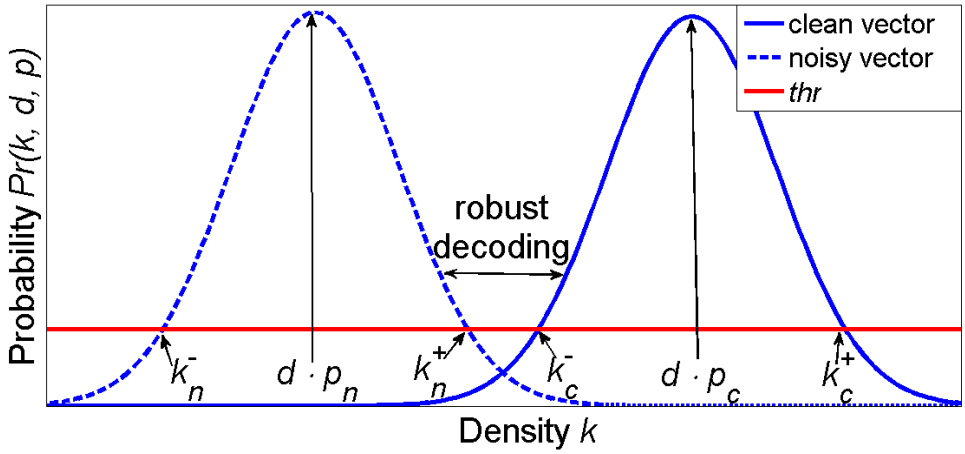


Figure 12: Explanation of vector's capacity. Solid line represents random HD-vector. Dashed line corresponds to noise introduced by majority sum.

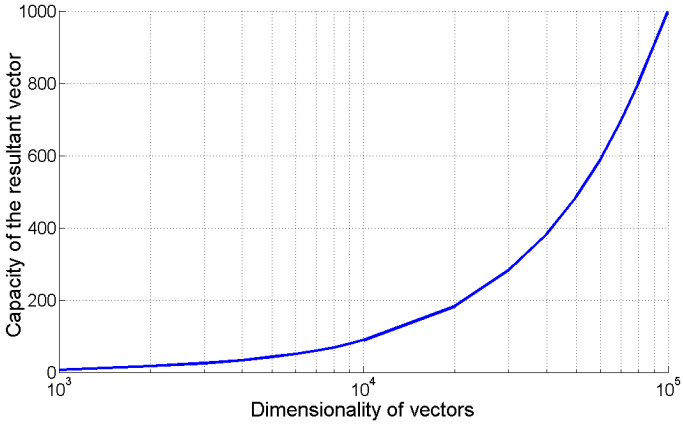


Figure 13: Capacity of the component vector versus the dimensionality. The calculations use a the threshold of thr = 10^{-6}

consisting of "1" and "-1" components was presented in [7]. The main difference between two methods is that the calculations in equation (9) require thr parameter, while the analysis in [7] requires the size of the item memory and the probability of successful decoding. Nevertheless, the two approaches for capacity estimation largely agree, for example, in [7] for dimensionality 90000 the capacity is 1000, the same capacity for (9) is achieved with dimensionality 100000.

7.3 Calculation of the Number of Common Component Vectors in Two Resulting Vectors

Given the rather conservative limits on the number of robustly decodable elements in a distributed representation it is important that the proposed HoloGN architecture can estimate the similarity between different patterns *without* decoding them. This subsection provides a method for quantitatively measuring the number of overlapping elements as a function of their relative normalized Hamming distance. Denote m and n as lengths of two patterns, $m <= n$, and denote c as the number of common elements in these patterns. Let $\mathbf{M}$ be a $c \times d$ matrix of common elements where each row contains a random HD-vector of dimension d encoding element c_i. Denote an arbitrary column of matrix $\mathbf{M}$ as $\mathbf{C}$. Since rows in $\mathbf{M}$ are independent, the density of **ones** in each column also follows the binomial distribution with $p = 0.5$ and length c. Denote the number of **ones** in column $\mathbf{C}$ as $||\mathbf{C}||_1$.

In order to calculate the normalized Hamming distance between the distributed representations of two patterns with known m, n and c, consider all possible cases when bits in the same position are different. The normalized Hamming distance between two patterns can be estimated with

$$\Delta_H = p(c,m,n) = \sum_{||\mathbf{C}||_1=0}^{c} \frac{\binom{c}{||\mathbf{C}||_1}}{2^c} \cdot (p_1(m,c,||\mathbf{C}||_1)\cdot \cdot p_0(n,c,||\mathbf{C}||_1)+p_0(m,c,||\mathbf{C}||_1)\cdot p_1(n,c,||\mathbf{C}||_1)); \tag{10}$$

where $p_i(j,c,||\mathbf{C}||_1)$ stands for the probability of having i (0 or 1), when the representation consists of $j = m$ or $j = n$ atomic vectors and c of these vectors are overlapped.

Due to the symmetry in the calculation of probabilities, $p_i(j,c,||\mathbf{C}||_1)$ is presented only for the case of $p_1(m,c,||\mathbf{C}||_1)$. There are three possible cases for calculation of $p_1(m,c,||\mathbf{C}||_1)$:

- if $||\mathbf{C}||_1$ is more than $m/2$, then the result of the majority sum is '1', i.e. p_1 is 1;
- if number of possible '1's is less than $m/2$, then probability of p_1 is 0;
- otherwise the probability should take into account all possible combinations, and their probabilities.

Equation (11) specifies probability for $p_1(m,c,||\mathbf{C}||_1)$, and following equation does the same for $p_0(m,c,||\mathbf{C}||_1)$:

$$p_1(m,c,||\mathbf{C}||_1) = \begin{cases} 1, \text{ when } ||\mathbf{C}||_1 > \frac{m}{2} \\ 0, \text{ when } (m-c) < \frac{m+1}{2} - ||\mathbf{C}||_1 \\ \dfrac{\sum_{i=(\frac{m+1}{2}-||\mathbf{C}||_1)}^{(m-c)} \binom{m-c}{i}}{2^{(m-c)}}, \text{ otherwise} \end{cases} \tag{11}$$

$$p_0(m, c, ||\mathbf{C}||_1) = \begin{cases} 1, \text{ when } (c - ||\mathbf{C}||_1) > \frac{m}{2} \\ 0, \text{ when } (m - c) < \frac{m+1}{2} - (c - ||\mathbf{C}||_1) \\ \dfrac{\sum_{i=(\frac{m+1}{2}-(c-||\mathbf{C}||_1))}^{(m-c)} \binom{m-c}{i}}{2^{(m-c)}}, \text{ otherwise} \end{cases} \tag{12}$$

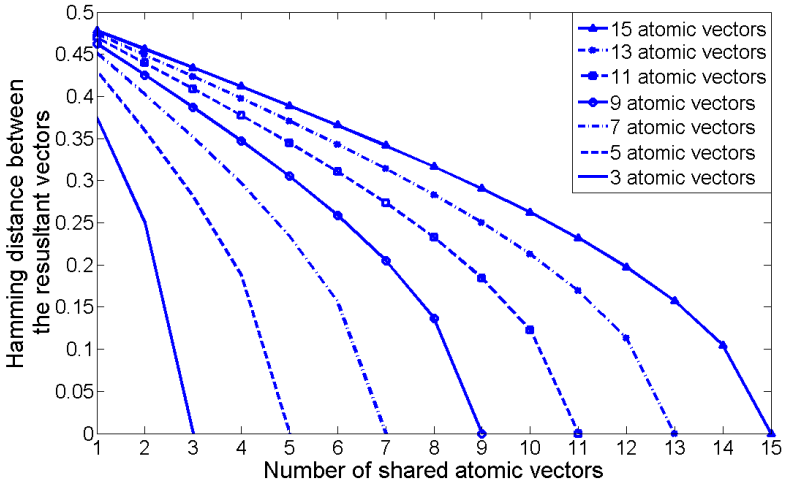


Figure 14: The normalized Hamming distance between two resulting vectors against number of components in common. The number of atomic vectors is the same, $m = n$.

Figure 14 shows the normalized Hamming distances between two resulting vectors for different numbers of overlapping vectors. The results show that the larger the number of common elements, the smaller the normalized Hamming distance between resulting vectors.

This method opens a way towards constructing and analyzing patterns far beyond VSA's robustly decodable capacity. The problem with practical application of this method, however, comes with the rapid convergence of the normalized Hamming distance indicator to 0.5, making the difference between analyzable patterns indistinguishable as illustrated in Figure 14. For example, for HoloGN representations of patterns with 15 elements, sub-patterns of 3 overlapped elements are robustly detected, while patterns with fewer overlapped elements are indistinguishable. Thus, the minimal number of overlapped elements in two patterns which can be robustly detected using the normalized Hamming distance indicator is called bundle's *sensitivity*.

The analysis of the sensitivity is similar to the analysis of the capacity of VSA representation in section 7.2. For two patterns of length m and n elements, and c overlapped

components, the sensitivity is calculated by

$$\begin{aligned} &\text{Sensitivity}(d, \text{thr}, m, n) = \\ &\min_c(k^+(d, p(c, n, m), \text{thr}) <= k^-(d, 0.5, \text{thr})) \end{aligned} \tag{13}$$

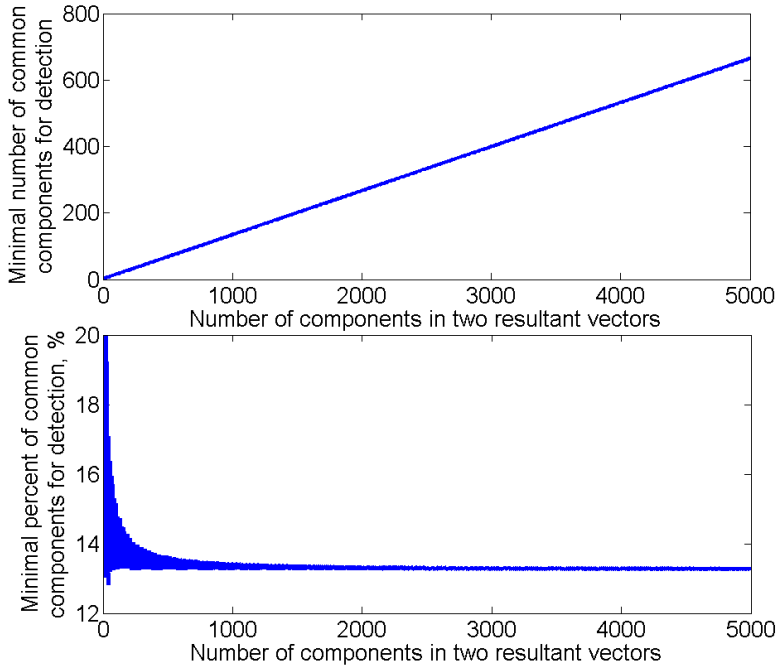


Figure 15: Minimal number of common components, which is sensible between two patterns of the same size against the size of patterns, $d = 10000$, $\text{thr} = 10^{-6}$.

Figure 15 demonstrates the development of the sensitivity threshold with the number of elements in the compared patterns. The results show that the number of components for robust detection grows linearly with the size of the pattern. Patterns with more than 500 elements should contain at least 14% overlapped elements to be robustly detected by the proposed method.

8 Conclusion

This article presented Holographic Graph Neuron - a novel approach for memorizing patterns of generic sensor stimuli. HoloGN is built upon the previous Graph Neuron algorithm and adopts a Vector Symbolic representation for encoding Graph Neuron's

states. The adoption of the Vector Symbolic representation ensures a single-layer design for the approach, which leads to much simpler computational operations. The approach presented in the paper possesses a number of unique properties. First, it enables a linear (with respect to the number of stored entries) time search for an arbitrary sub-pattern. Second, while maintaining the previously reported properties of the Hierarchical Graph Neuron, HoloGN improves the noise resistance of the architecture leading to substantial improvement of pattern recall accuracy.

Acknowledgements

This work is partially supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), institutional grant IG2011-2025.

References

[1] E. Osipov, A. I. Khan, and A. Anang, "Holographic graph neuron," in *Computer and Information Sciences (ICCOINS), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[2] B. B. Nasution and A. I. Khan, "A hierarchical graph neuron scheme for real-time pattern recognition," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 212–229, February 2008.

[3] A. I. Khan and A. H. M. Amin, "One-shot associative memory method for distorted pattern recognition," in *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia*, vol. 4830, 2007, pp. 705–709.

[4] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, October 2009.

[5] S. D. Levy and R. Gayler, "Vector symbolic architectures: A new building material for artificial general intelligence," in *Proceedings of the 2008 Conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008, pp. 414–418. [Online]. Available: http://dl.acm.org/citation.cfm?id=1566174.1566215

[6] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[7] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Computation*, vol. 25, no. 8, pp. 2038–2078, 2013.

[8] R. Zbikowski, "Fly like a fly," *IEEE Spectrum*, vol. 42, no. 11, pp. 46–51, November 2005.

[9] Y. M. Song, Y. Xie, V. Malyarchuk, J. Xiao, I. Jung, K.-J. Choi, Z. Liu, H. Park, C. Lu, R.-H. Kim, R. Li, K. B. Crozier, Y. Huang, and J. A. Rogers, "Digital cameras with designs inspired by the arthropod eye," *Nature*, vol. 497, no. 7447, pp. 95–99, May 2013.

[10] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: A tutorial and survey," *IEEE journal of solid-state circuits*, vol. 41, no. 3, pp. 712–727, 2006.

[11] D. Perrett, E. Rolls, and W. Caan, "Visual neurones responsive to faces in the monkey temporal cortex," *Experimental Brain Research*, vol. 47, no. 3, pp. 329–342, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF00239352

[12] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, 1988.

[13] H. Meng, K. Appiah, A. Hunter, S. Yue, M. Hobden, N. Priestley, P. Hobden, and C. Pettit, "A modified sparse distributed memory model for extracting clean patterns from noisy inputs," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, June 2009, pp. 2084–2089.

[14] D. Rachkovskij, E. Kussul, and T. Baidyk, "Building a world model with structure-sensitive sparse binary distributed representations," *Biologically Inspired Cognitive Architectures*, vol. 3, pp. 64–86, 2013.

[15] J. T. Abbott, J. B. Hamrick, and T. L. Griffiths, "Approximating bayesian inference with a sparse distributed memory system."

[16] P. Kanerva, "Fully distributed representation," in *Real world computing symposium*, 1997, pp. 358–365.

[17] D. Kleyko, E. Osipov, N. Papakonstantinou, V. Vyatkin, and A. Mousavi, "Fault detection in the hyperspace: Towards intelligent automation systems," in *IEEE International Conference on Industrial Informatics, INDIN*, 2015, pp. 1–6.

[18] D. Kleyko and E. Osipov, "Brain-like classifier of temporal patterns," in *The Proceeding of the 2nd International Conference on Computer and Information Sciences - ICCOINS*, 2014, pp. 1–6.

[19] K. J. Schultz, *Content Addressable Memory*. N. T. Limited, 1999.

[20] P. Kanerva, "A family of binary spatter codes," in *ICANN '95: International Conference on Artificial Neural Networks*, 1995, pp. 517–522.

[21] D. Rasmussen and C. Eliasmith, "A neural model of rule generation in inductive reasoning," *Topics in Cognitive Science*, vol. 3, no. 1, pp. 140–153, 2011.

[22] B. Emruli, R. W. Gayler, and F. Sandin, "Analogical mapping and inference with binary spatter codes and sparse distributed memory," in *The proceedings of the IJCNN*, 2013, pp. 1–8.

[23] D. Kleyko, E. Osipov, R. W. Gayler, A. I. Khan, and A. G. Dyer, "Imitation of honey bees' concept learning processes using vector symbolic architectures," *Biologically Inspired Cognitive Architectures*, vol. 14, pp. 57–72, 2015.

[24] D. Kleyko, E. Osipov, M. Bjork, H. Toresson, and A. Oberg, "Fly-The-Bee: A Game Imitating Concept Learning in Bees," *Procedia Computer Science*, vol. 71, pp. 25–30, 2015.

[25] S. Levy, S. Bajracharya, and R. Gayler, "Learning behavior hierarchies via high-dimensional sensor projection."

[26] D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, "Dependable mac layer architecture based on holographic data representation using hyper-dimensional binary spatter codes," in *Multiple Access Communications : 5th International Workshop, MACOM 2012*. Springer, 2012, pp. 134–145.

[27] P. Jakimovski, H. R. Schmidtke, S. Sigg, L. W. F. Chaves, and M. Beigl, "Collective communication for dense sensing environments," *Journal of Ambient Intelligence and Smart Environments (JAISE)*, vol. 4, no. 2, pp. 123–134, March 2012.

[28] M. Gardner, "Mathematical games," *Scientific American*, vol. 223, no. 4, pp. 120–123, October 1970.

[29] T. Plate, *Distributed Representations and Nested Compositional Structure*. University of Toronto, PhD Thesis, 1994.

[30] D. Kleyko and E. Osipov, "On bidirectional transitions between localist and distributed representations: The case of common substrings search using vector symbolic architecture," *Procedia Computer Science*, vol. 41, pp. 104–113, 2014.

Paper C

On bidirectional transitions between localist and distributed representations: The case of common substrings search using Vector Symbolic Architecture

Authors:
Denis Kleyko and Evgeny Osipov

Reformatted version of paper originally published in:
Procedia Computer Science.

On bidirectional transitions between localist and distributed representations: The case of common substrings search using Vector Symbolic Architecture

Denis Kleyko and Evgeny Osipov

Abstract

The contribution of this article is twofold. First, it presents an encoding approach for seamless bidirectional transitions between localist and distributed representation domains. Second, the approach is demonstrated on the example of using Vector Symbolic Architecture for solving a problem of finding common substrings. The proposed algorithm uses elementary operations on long binary vectors. For the case of two patterns with respective lengths $L1$ and $L2$ it requires $\Theta(L1+L2-1)$ operations on binary vectors, which is equal to the suffix trees approach – the fastest algorithm for this problem. The simulation results show that in order to be robustly detected by the proposed approach the length of a common substring should be more than 4% of the longest pattern.

1 Introduction

Distributed data representation is widely used in the area of cognitive computing for representing and reasoning upon semantically bound information [1], [2]. The transitions between localist representations are normally done only during the encoding of concepts and their later recalling from the item-memory. However, there is a limit on the number of individual concepts being represented in one distributed representation after which the recalling becomes a tedious and an error prone task [3]. That is why once encoded, all operations e.g., generalization and reasoning, are performed in the domain of distributed representations only. There are, however, cases when the transition back to the localist representation is essential for interpreting the results of intermediate operations with the distributed representation. This is specifically needed if the distributed representation is used to characterize complex systems [4], e.g. like industrial processes. This article presents an encoding approach, which allows seamless bidirectional transitions between localist and distributed representation domains. The approach is demonstrated on the example of using Vector Symbolic Architecture for solving a problem of finding common substrings.

Searching common substrings is a typical task in processing of strings of symbols. A typical application is the search for plagiarism in two texts. The problem is exemplified

in Figure 1. There given two strings of characters of different lengths four common substrings are identified and the longest common substring contains three symbols.

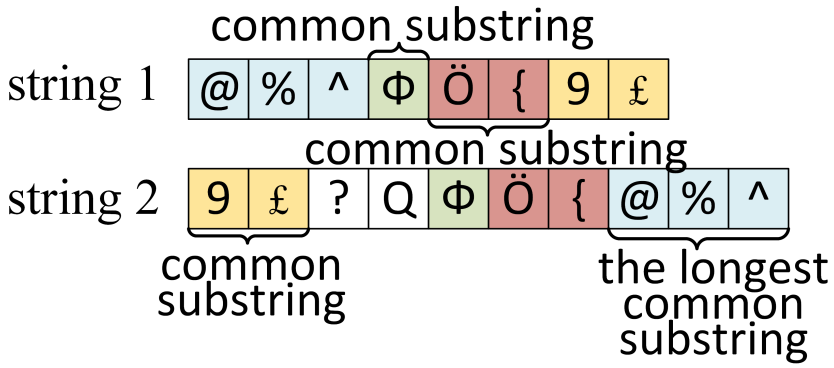


Figure 1: An example of common substrings in two strings.

The contribution of this article is twofold. First, it presents an encoding approach for seamless bidirectional transitions between localist and distributed representation domains. Second, this article presents a solution for typically localist problem with the performance comparable to traditional algorithms.

The article is structured as follows. Section 2 presents an overview of the related work relevant to the context of this article. Section 3 provides the fundamentals of the theory of Vector Symbolic Architecture and Binary Spatter Codes relevant to the scope of the article. Section 4 presents the main contribution of this article – VSA based representation and search of the common substrings. The performance evaluation of the proposed approach is given in Section 5. The conclusions are presented in Section 6.

2 Related Work

Algorithms for strings analysis address a wide range of problems. The problem of exact string matching is a common task for instance in web search. There are several algorithms for addressing this problem. The most known examples of such algorithms are Knuth-Morris-Pratt algorithm [5], Rabin-Karp algorithm [6], Boyer-Moore algorithm [7], many more could be found in [8]. This article considers a more general problem of finding common substrings in two arbitrarily long strings with lengths $L1$ and $L2$. This problem for example has several applications in computational biology [9]. The brute-force solution to this problem has $\Theta(L^3)$ complexity, where L is the length of the longest string. The algorithms based on dynamic programming [10] has $\Theta(L1 \cdot L2)$ computational com-

plexity. An approach using suffix trees [11], [12] solves the longest common substring problem using $\Theta(L1 + L2)$ operations, i.e. in linear time. The proposed in this article approach has the complexity of the suffix tree based algorithms.

Vector Symbolic Architecture is a bio-inspired representation of semantically bound information. Similarly to brain activity where simple mental events involve the simultaneous activities of very large number of dispersed neurons [1]. Information in VSA is represented distributively, where a single concept is associated with multiple codes. In VSA this is achieved by using codewords of very large dimension. There are several different types of VSA using different representations, e.g. [2], [1], [13], [14], [15]. This article utilizes a subclass of VSA based on so-called Binary Spatter Codes (BSC) [3].

In the context of the problem of strings analysis VSA was previously used in [16], [17], [18] for estimating the Levenshtein distance, a metric for measuring the difference between two strings. In essence these methods perform the q-gram analysis of the sequences and use VSA for distributed representation of the analysis results and subsequent comparison. Recent applications of VSA for analysis of generic patterns are proposed in [19], [20].

3 Fundamentals of Vector Symbolic Architecture and Binary Spatter Codes

Vector Symbolic Architecture is an approach for encoding and operations on distributed representation of information. VSA is so far mainly used in the area of cognitive computing for representing and reasoning upon semantically bound information [1], [2].

The fundamental difference between distributed and localist representations of data as follows. In traditional (localist) computing architectures each bit and its position within a structure of bits are significant. For example a field in a database has a predefined offset amongst other fields and a symbolic value has unique representation in ASCII codes. In the distributed representation all entities are represented by binary random vectors of very high dimension also called Binary Spatter codes. Further in the article for brevity reasons *HD-vector* term is used when referring to BSC codes. *High dimensionality* means here several thousand of binary positions for representing a single entity. In [1] it is proposed to use vectors of 10000 binary elements.

Randomness means that the values on each position of an HD-vector are independent of each other, and "0" and "1" components are equally probable, $p_0 = p_1 = 0.5$. On very high dimensions, the distances from any arbitrary chosen HD-vector more than 99.99 % of all other vectors in the representation space are concentrated around 0.5 Hamming distance. Interested readers are referred to [1] and [21] for comprehensive analysis of probabilistic properties of the hyperdimensional representation space.

3.1 Similarity metric

A similarity between two binary representation is characterized by Hamming distance, which measures the number of positions in two compared vectors in which they are

different:

$$\Delta_H(A,B) = \frac{\|A \otimes B\|_1}{n} = \frac{\sum_{i=0}^{n-1} a_i \otimes b_i}{n},$$

where a_i, b_i are bits on positions i in vectors A and B of dimension n and $\otimes$ denotes the bit-wise XOR operation.

3.2 Generation of HD-vectors

Random binary vectors with the above properties could be generated based on Zadoff-Chu sequences [22], a method widely used in telecommunications to generate pseudo-orthogonal preambles. Using this principle a sequence of K pseudo-orthogonal vectors to a given initial random HD-vector A is obtained by cyclically shifting A on i positions, where $1 < i \leq K$. Further in the article this operation is denoted as $Sh(A,i)$. Cyclic shift operation has the following properties:

- Cyclic shift is invertible, i.e. if $B = Sh(A,i)$ then $A = Sh(B,-i)$;
- Cyclic shift is associative in the sense that $Sh(B,i+j) = Sh(Sh(B,i),j) = Sh(Sh(B,j),i)$;
- It preserves Hamming weight of the result: $\|B\|_1 = \|Sh(B,i)\|_1$;
- The result is dissimilar to the vector being shifted: $\frac{\|B \otimes Sh(B,i)\|_1}{n} = 0.5$.

Note that the cyclic shift is a special case of the permutation operation [1]. In the context of VSA permutations were previously used to encode sequences of semantically bound elements. The associativity property of the cyclic shift operation as stated above was not used in the context of VSA architectures since the task of aggregate transformation, e.g. comparison of the similarity between two relatively displaced sequences of elements is not relevant in current VSA applications. Thus, the cyclic shift based encoding scheme presented in this article represents the novel usage of the previously known operation in the considered application.

3.3 Bundling of vectors

Joining several entities into a structure is done by a bundling operation. It is implemented by a thresholded sum of the HD-vectors representing entities. A bit-wise thresholded sum of K vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. Further terms "thresholded sum" and "MAJORITY sum" are used interchangeably and denoted as $[A+B+C]$. The relevant properties of the MAJORITY sum are:

- The result is a random vector, i.e. the number of “1” components is approximately equal to the number of “0” components;

- The result is similar to all vectors included in the sum;
- Number of vectors involved into MAJORITY sum must be odd;
- The more HD-vectors involved into MAJORITY operations the closer is Hamming distance between the resultant vector and any HD-vector component to 0.5.
- If several copies of any vector are included into MAJORITY sum the resultant vector is closer to the dominating vector than to other components.

The algebra on VSA includes other operations e.g., binding, permutation. Since in the scope of this article they are not used the description of their properties is omitted.

4 VSA based search of the common substrings

This section presents the two main contributions of the paper: an encoding approach for seamless bidirectional transitions between localist and distributed representation domains and its application for solving the longest common substring problem. In simple words in the proposed approach patterns are firstly represented in a distributed way. Hamming distance is used as a similarity metric between patterns' representations the longest string is choses as a base line. The cyclic shift of the distributively represented shorter string is used instead for checking of all possible substrings combination in the localist representation. For each shift transformation Hamming distance between the longest string and the shifted string is measured. is recorded along with the shift value. The tuple (shift position, Hamming distance) for each shift value is stored in the list sorted in the descending order. After shifting along along all base line string's elements the minimal Hamming distance amongst all shifts is determined. This value indicates the presence of the longest common substring when the shorter string is shifted on the corresponding number of positions.

Suppose an alphabet of localist symbols is known and finite. Suppose also dictionary D_{HD} of random HD-vectors is generated for each symbol in the localist alphabet. Recall that the task is formulated as finding common substrings in two given strings with lengths $L1$ and $L2$ respectively.

The proposed approach consists of three phases:

- Encoding of the strings into VSA distributed data representation.
- Search of the shift positions in which strings have common substrings.
- Extraction of the common substrings.

The following subsections present a detailed description of these phases.

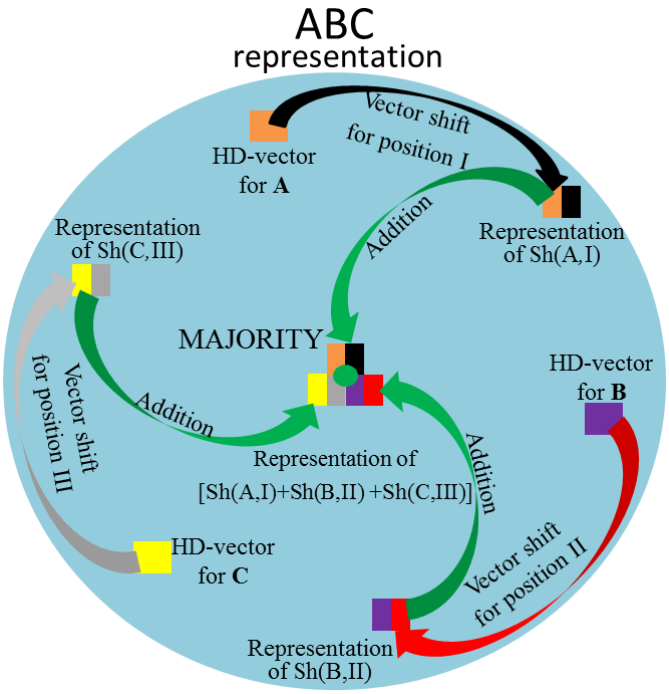


Figure 2: Illustration of the idea for VSA representation of a pattern by cyclic shift and MAJORITY sum of individual HD-vectors. As an example of the encoding scheme consider the string of 3 letters ABC. The VSA based representation of the string is constructed as follows: $P_{ABC} = [Sh(A, \mathtt{I}) + Sh(B, \mathtt{II}) + Sh(C, \mathtt{III})]$.

4.1 Forming VSA representation of patterns

Recall the property of the MAJORITY sum on similarity of the result to the dominating vector. The idea with encoding strings into the distributed representation is to use a MAJORITY sum of the distributes representations of the individual characters. Since strings of characters naturally contain repeating elements, in order to avoid similarity between two distributed representations of strings with the same elements on different positions, it is necessary to represent the same characters on different positions by orthogonal HD-vectors. The core of the proposed encoding scheme is to cyclically shift the initial HD-vector for a given symbol on the value of its position in the considered string. More formally a symbol S on position i is encoded as: $Sh(D_{HD}[S_i], i)$.

Then the VSA representation of the particular pattern is a MAJORITY sum of the distributed representation of individual elements of the pattern. Recall that MAJORITY sum operation produces a HD-vector, which is similar to the all HD-vectors-components. Schematically the idea of encoding the localist string in its distributed representation

is illustrated in Figure 2, where a blue circle is 2D projection of the hyperdimensional space.

Algorithm 1 presents a high-level logic for the VSA-based string encoding. Now when VSA based representations of strings is specified the next subsection presents procedure for the search of the common substrings.

Data: Pattern to represent: P; dictionary of HD-vectors: D_HD.
Result: Distributed representation of the pattern: P_HD.
`/* Dimensionality of HD-vectors */`
dimensionality:=10000;
`/* Creating an empty vector for the VSA representation */`
P_HD:=zeros(1,dimensionality);
`/* Forming representation of the pattern */`
for *(position:=1; position1<=length(P); position++)* **do**
  `/* Picking HD-vector for the current symbol from dictionary */`
  HDvect:=D_HD(P[position]);
  `/* Shifting HD-vector for a symbol on the current position */`
  HDvect_shifted:=Sh(HDvect, position);
  `/* Adding shifted vector to the representation */`
  P_HD:=P_HD + HDvect_shifted;
end
`/* Applying MAJORITY on the representation */`
P_HD:=MAJORITY(P_HD);

Algorithm 1: An algorithm for forming distributed representation of patterns.

4.2 The search of the common substrings in distributed representations

The common substrings search procedure in the distributed representation is straightforward and utilizes the following properties of the hyperdimensional space:

- If two strings have several individual elements in common, the Hamming distance between their distributed representations is less than 0.5;
- The larger number of overlapping in the two strings elements exist the closer is the Hamming distance to zero;
- Associativity of the cyclic shift operation enables the shift transformation of the string. Thus thanks to the associativity of the cyclic shift operation the distributed representation of the pattern can be changed applying only one operation.

The principle of the search procedure is explained on an example illustrated in Figure 3 and its high level logic is presented in Algorithm 2. There are two input strings to

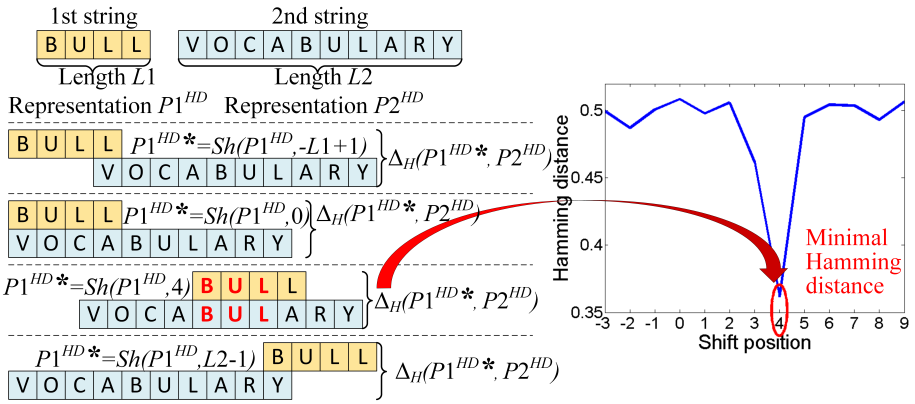


Figure 3: Illustration of search principles in the distributed domain.

compare, string $P1$="bull" and string $P2$="vocabulary". The distributed representations for initial strings have the following forms:

$$P1^{HD} = [Sh(B,1) + Sh(U,2) + Sh(L,3) + Sh(L,4)];\ P2^{HD} = [Sh(V,1) + Sh(O,2) + Sh(C,3) + Sh(A,4) + Sh(B,5) + Sh(U,6) + Sh(L,7) + Sh(A,8) + Sh(R,9) + Sh(Y,10)].$$

The longest sting is chosen as a base line. The search is implemented using only two operations:

- Cyclic shift of the representation $P1^{HD}$ of the shortest string;
- Calculation of the Hamming distance between the first shifted representation $P1^{HD*}$ and the second representation $P2^{HD}$.

In total one needs to perform $L1+L2-1$ shifts in order to account all possible relative positions of two strings. The range of possible shift transformations consists of positions between $-L1+1$ (the first case in Figure 3) and $L2-1$ (the last case in Figure 3).

Graphically it can be illustrated as a shift of the first initial string with respect to the second initial string. In the example when applying shift on the 4th position the shifted strings will take form:

$$P1^{HD*} = Sh(P1^{HD},4) = Sh([Sh(B,1) + Sh(U,2) + Sh(L,3) + Sh(L,4)],4) = [Sh(B,5) + Sh(U,6) + Sh(L,7) + Sh(L,8)].$$

Since the shifted representation $P1^{HD*}$ clearly has common elements with $P2^{HD}$, the Hamming distance between two strings will be the smallest as the two strings contain the largest number of the overlapping characters.

Data: Length of patterns: $L1$, $L2$; Distributed representation for both patterns: $P1\_HD$, $P2\_HD$.
Result: Shift position, which leads to the smallest Hamming distance between patterns representations: min_shift

```
/* Calculating Hamming distance for possible combinations                */
for (shift:=-L1+1; shift<=L2-1; shift++) do
    /* Shifting the 1st pattern representation on the current shift      */
    P1_HD_shifted:=Sh(P1_HD, shift);
    /* Calculating Hamming distance between representations              */
    HDist[shift]:=HammingDistance(P2_HD, P1_HD_shifted);
end
/* Arranging Hamming distance in the ascending order                     */
HDist:=SORT(HDist);
/* The first element in HDist is a shift for the longest common substring
   */
```

Algorithm 2: A procedure for the search of shift position of pattern representation with the common substrings. The output of the algorithm is sorted Hamming distances and corresponding shift positions. Shift with the minimal Hamming distance contains the longest common substring.

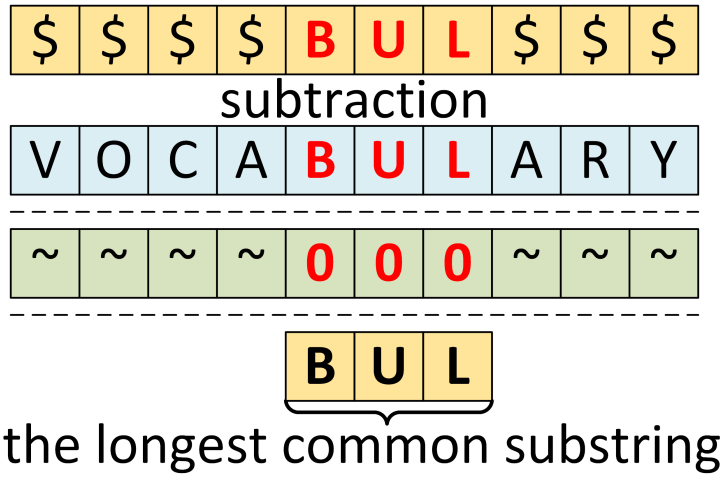


Figure 4: An illustration of extraction of the longest common substring.

4.3 Extraction of the common substrings

The third phase of extracting the common is straight forward and happens in the localist domain, thus avoiding the need for decoding for individual elements from the distributed

representation.

Figure 4 gives an intuitive illustration of this process. For the extraction the shortest localist string is shifted on the number of positions for which the minimal Hamming distance is found (in the considered example it is position number 4). The two strings are subtracted and the task now is to find the longest sequence of zeros, which is the mask for the common substring.

5 Illustrative performance and discussion

5.1 Computational complexity of the approach

This sections investigates performance of the proposed approach. The following aspects are addressed: computational complexity, scalability and limitations.

Current hardware architectures operate on sizes of buffers equal to 32 or 64 bits. However, hyperdimensional computing in turn uses vectors of very high dimensionality more than 1000 elements. Consider the case when the hardware operating with such vectors is available and it could calculate such operations as cyclic shift, XOR, MAJORITY sum and Hamming distance in one clock cycle. This assumption is feasible according to [23]. With this assumption one can see that both Algorithm 1 and Algorithm 2 have maximal size of loop equal to $L1 + L2 - 1$, this statement is also relevant to the extraction of the common substring when the shift is known. Thus the overall computational complexity of the proposed approach is $\Theta(L1 + L2 - 1)$, which is the same as the complexity of the suffix trees approach and better than for the standard approach using dynamic programming with $\Theta(L1 \cdot L2)$ operations.

5.2 Simulation of the approach

The main limitations of the proposer approach stem from the nature of the distributed representation. Namely amount of vectors, which can be merged into a single representation. Firstly, the robustness of the algorithm depends on the number of vectors involved into MAJORITY operation, because it affects the similarity between the resultant vector and initial ones. In other words it is not the same in terms of Hamming distance when the resultant vector consists of 10 or 100 HD-vectors. In the second case the Hamming distance is very close to 0.5. Similarly Hamming distance to the bundle of several initial HD-vectors more similar to the resultant vector than only one initial vector. Another restriction of the approach is its sensitivity i.e., the minimal length of the common substring, which can be robustly detected.

To address these limitations two simulations were performed. In the first simulation the interval between the shift for the longest common substring and the second closest alternative was investigated. For the simulation the length of the second string was fixed to 100 elements. The first string was changed from 1 element of the second string to all 100 elements of the second string. Figure 5 presents the simulation results.

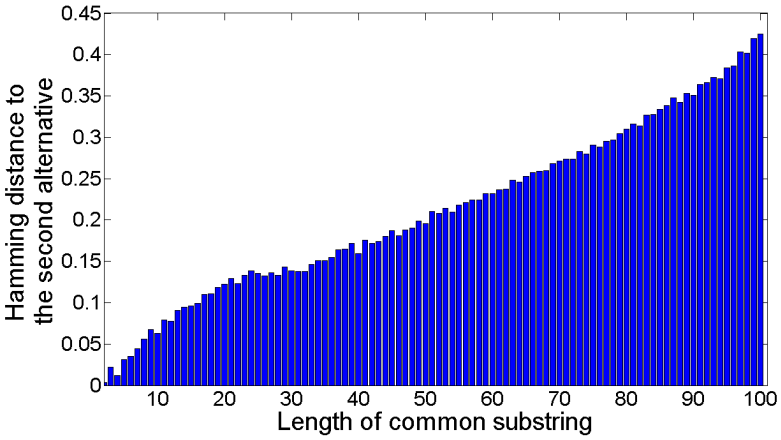


Figure 5: Hamming distance to the second alternative.

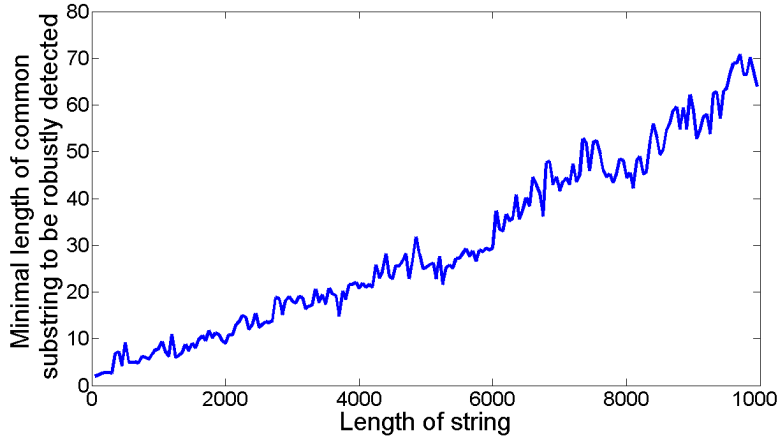


Figure 6: Sensitivity of the approach against the length of string.

One can see that the robustness of the algorithm increases with the increase in the length of the common substring. The results for the second simulations are illustrated in Figure 6. They present the minimal length of the common substring, which is detected correctly against the length of the second string. The results give a practical restriction, in order to be robustly detected the length of the common substring should be more than 4% of the longest string.

5.3 On search of the longest substring in a noisy string

One problem, which is not considered so far is the case where the compared strings contain common elements, which do not form substring. An example of such case is the following strings ABABABAB123 and ACACACAC*123. In this case the presented above algorithm would stop after finding sequences of four 'A' elements. This, however, is no a substring, which in the example above must be '123'. This problem can be resolved by collecting statistics of Hamming distances for each iteration in Algorithm 2. In this case when transiting back into the localist domain (subsection 4.3) one should consider the top N shift positions for the common substring. The evaluation of the optimal number of candidates is a part of the future development of the proposed solution.

6 Conclusion

This paper proposed an encoding approach for seamless bidirectional transitions between localist and distributed representation domains. The approach was demonstrated on the example of using Vector Symbolic Architecture for solving the problem of finding common substrings. The complexity of the proposed approach for the common substrings search in terms of vector operations is comparable with the traditional algorithms. The simulation results provided a practical restriction to the approach that for the robust detection the length of the common substring should be more than 4% of the longest string.

Acknowledgements

This work is partially supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), institutional grant IG2011-2025.

References

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[2] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[3] P. Kanerva, "Fully distributed representation," in *Real world computing symposium*, 1997, pp. 358–365.

[4] B. Emruli, F. Sandin, and J. Delsing, "Vector space architecture for emergent interoperability of systems by learning from demonstration," *Biologically Inspired Cognitive Architectures*, vol. 11, pp. 53–64, 2015.

[5] D. Knuth, J. H. Morris, and V. Pratt, "Fast pattern matching in strings," *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977.

[6] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987.

[7] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[8] B. Melichar, J. Holub, and T. Polcar, *Text searching algorithms. Forward string matching*, 2005.

[9] I. Alsmadi and M. Nuser, "String matching evaluation methods for DNA comparison," *International Journal of Advanced Science and Technology*, vol. 47, pp. 13–32, 2012.

[10] D. Gusfield, *Algorithms on strings, trees, and sequences.* Cambridge University Press, 1997.

[11] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, 1995.

[12] R. Giegerich and S. Kurtz, "From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction," *Algorithmica*, vol. 19, no. 3, pp. 331–353, 1997.

[13] R. Gayler, "Multiplicative binding, representation operators & analogy," in *Gentner, D., Holyoak, K. J., Kokinov, B. N. (Eds.), Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*, New Bulgarian University, Sofia, Bulgaria, 1998, pp. 1–4.

[14] D. A. Rachkovskij and E. M. Kussul, "Binding and normalization of binary sparse distributed representations by context-dependent thinning," *Neural Computation*, vol. 13, no. 2, pp. 411–452, 2001.

[15] D. Aerts, M. Czachor, and B. D. Moor, "Geometric analogue of holographic reduced representation," *Journal of Mathematical Psychology*, vol. 53, pp. 389–398, 2009.

[16] I. Misuno, D. Rachkovskij, S. Slipchenko, and A. Sokolov, "Searching for text information with vector representations," *Journal of Problems in Programming*, vol. 4, pp. 50–59, 2005.

[17] A. Sokolov and D. Rachkovskij, "Approaches to sequence similarity representation," *Journal of Information Theories and Applications*, vol. 13, no. 3, pp. 272–278, 2005.

[18] A. Sokolov, "Vector representations for efficient comparison and search for similar strings," *Cybernetics and Systems Analysis*, vol. 43, no. 4, pp. 484–498, 2007.

[19] E. Osipov, A. Khan, and A. M. Amin, "Holographic graph neuron," in *The Proceeding of the 2nd International Conference on Computer and Information Sciences - ICCOINS*, 2014, pp. 1–6.

[20] D. Kleyko and E. Osipov, "Brain-like classifier of temporal patterns," in *The Proceeding of the 2nd International Conference on Computer and Information Sciences - ICCOINS*, 2014, pp. 1–6.

[21] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, 1988.

[22] B. Popovic, "Generalized chirp-like polyphase sequences with optimum correlation properties," *Information Theory, IEEE Transactions on*, vol. 38, no. 4, pp. 1406–1409, 1992.

[23] I. S. Haque, V. S. Pande, and W. P. Walters, "Anatomy of high-performance 2D similarity calculations," *Journal of Chemical Information and Modeling*, vol. 51, no. 9, pp. 2345–2351, 2011.

Paper D

Fault Detection in the Hyperspace: Towards Intelligent Automation Systems

Authors:
Denis Kleyko, Evgeny Osipov, Nikolaos Papakonstantinou, Valeriy Vyatkin, and Arash Mousavi

Reformatted version of paper originally published in:
IEEE International Conference on Industrial Informatics (INDIN), 2015, Cambridge, UK.

Fault Detection in the Hyperspace: Towards Intelligent Automation Systems

Denis Kleyko, Evgeny Osipov,
Nikolaos Papakonstantinou, Valeriy Vyatkin, and Arash Mousavi

Abstract

This article presents a methodology for intelligent, biologically inspired fault detection system for generic complex systems of systems. The proposed methodology utilizes the concepts of associative memory and vector symbolic architectures, commonly used for modeling cognitive abilities of human brain. Compared to classical methods of artificial intelligence used in the context of fault detection the proposed methodology shows an unprecedented performance, while featuring zero configuration and simple operations.

1 Introduction

This article considers intelligent fault detection and identification in generic complex system of systems. Examples of such systems include modern automated mechatronic systems in different industrial processes as well as modern ICT systems with virtualized architectures. The common feature of such system is hard dependability requirements on the one hand and their continuously growing complexity on the other. Fault detection and identification in such systems is continuously critical research area. An additional challenge for future fault detection systems is distribution of the system across networked hardware devices. Distributed automation will enable flexible and intelligent industries. The distribution of intelligence is currently empowered by several standards and technologies on different levels. The IEC 61499 architecture [1] has been conceived to facilitate the use of distributed automation intelligence. While the first industrial applications [2] of commercial IEC 61499 compliant tools and platforms show the standards benefits in terms of improved design performance, by itself it does not address the complexity of the processes at the management layer. This calls for the need of enhancing the architecture with functions for autonomous management. Recent advances in merging the IEC 61499 architecture with multi-agent systems (MAS) [3, 4, 5] show dramatic enhancement of the system's autonomy. A principle approach for one the proposed architectures is visualized in Fig. 1.

At the same time condition monitoring and intelligent maintenance of industrial systems is becoming increasingly important function of automation systems, with the growing application of the artificial intelligence methods. One of the hottest trends in artificial intelligence is to enable on-line learning and interpretation of heterogeneous data streams. The main shift of the paradigm here is the transition from traditional methods for data

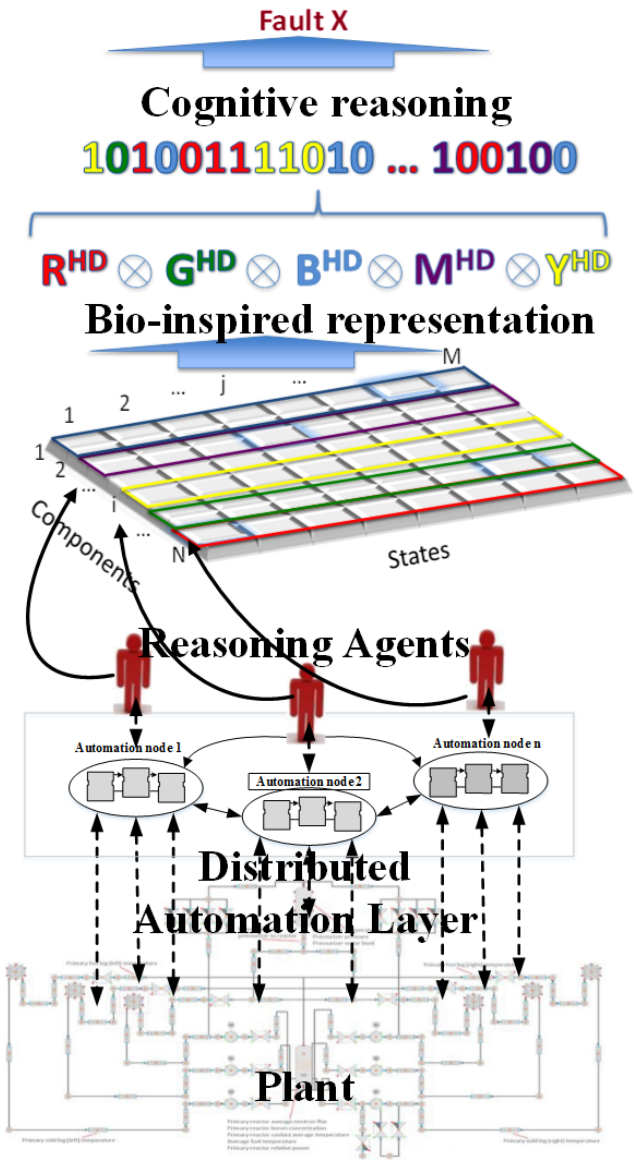


Figure 1: A distributed automation system enhanced with the autonomous intelligence capabilities based on the usage of multi-agent technology and VSA representation.

analysis and visualization, which require substantial manual work and knowledge engineering, to purely online autonomous learning of streaming data. The core idea with

the online learning is to create an *on-line model* of temporal patterns in data streamed by multiple heterogeneous sources (sensors, actuators, controllers, etc.). In this way the model will evolve with the evolution of the underlying process. Such modeling method would address one of the main challenges of the current control systems – the outdating of underlying plant models – and in turn enable more efficient fault management.

In light of the above this article proposes a biologically inspired approach for fault detection, which has a potential for distributed implementation with IEC 61499 and MAS. The approach adopts a mathematical framework previously used for modeling of human cognitive abilities called Vector Symbolic Architectures (further referred to as VSA) [6]. In simple words the proposed approach aims to learn and generalize temporal patterns in streams of telemetrical data generated across the plant during its stable and/or faulty situation operation. The created in this way on-line model keeps getting updated as the particular plant or automation system evolves. To this end the proposed approach is suggested as an overlay layer on top of the existing fault detection and management systems, which would enhance the robustness of decision making of the overall system. The proposed method demonstrates the best accuracy (up to 96%) of fault detection compared to the results from applying classical methods: neural networks, K-nearest neighbors (KNN) and decision trees.

The article is organized as follows. Section 2 presents an overview of the related work. Section 3 outlines the methodology. Section 4 presents the main contribution of the article: it introduces the essentials of used theories and presents the proposed methodology for fault detection in complex systems. The results of benchmarking with the related approaches are presented in Section 5. The discussion of the implications and conclusions follow in Section 6.

2 Literature review

This section contains a summary of the state of the art in fault detection and diagnosis research. The efforts in this research domain are extensive and many alternative fault detection methods and hybrid configurations have been proposed.

The machine learning research domain focuses on the development of algorithms which can enable computers to learn from data. Even though it has been a very active research area since the development of the first artificial intelligence applications, it has received a boost of interest because of the increased computational resources and the development of new machine learning methods [7] with many engineering applications.

The plethora of research on the fault detection and diagnosis domain has motivated papers with overview of state of the art. The trilogy of publications [8, 9, 10] contains a summary of fault detection literature and classifies the research efforts into three categories, depending on the algorithm, which is used. The "quantitative model-based" methods are based on system models to analytically identify differences between the expected and actual behavior. Decision rules are then applied to perform the fault identification and diagnosis [8]. The "Qualitative models and search strategies" category includes methods based on qualitative system models, like topographic templates cre-

ated using expert knowledge and fault trees [9]. The third category of fault detection and identification methods is the "Process history based methods", which do not utilize any knowledge about the system's structure, but rely on data sets of simulated or real process data as input to quantitative (e.g. Artificial Neural Networks) or qualitative (e.g. expert systems) methods [10]. In [11] a literature review, which focuses on model based methods is presented, the [12] contains an overview of artificial intelligence based methods.

Artificial Neural Network (ANN) based systems have been proposed for safety critical applications [13] and for the development of quantitative fault diagnosis and identification systems.

Hybrid systems involving ANNs and other methods have also been suggested for fault identification applications. An ANN and an analytical method were combined into a hybrid diagnostic system applied on a NPP case study. A dynamic neuro-fuzzy network and a dynamic ANN is used in [14] to create an advisory system for NPP accident diagnosis. Simple time-series and data pre-processed with Fast Fourier Transformation are used in [15] to train ANNs for improved fault identification and for tolerance against measurement drifts. ANNs were also combined with statistical control charts for fault identification.

Another quantitative method used for fault diagnosis is based on decision trees. Fault identification systems, which use decision trees have been applied on AC transmission lines, photovoltaic arrays , power systems [16] and migration paths to decision trees from fault trees for fault identification have been proposed for the International Space Station.

Optimized fuzzy clustering, Data-driven modeling [17] and residual space analysis, hidden Markov models, Independent Component Analysis and Dynamic Case Based Reasoning based methods were also proposed for fault identification.

Distributed approach to fault detection through allocation of computationally demanding tasks between different sensor nodes is proposed in [18]. Another decentralized approach for fault diagnosis is presented in [19]. Authors propose multiblock kernel partial least squares approach.

Associative memory (AM) is a sub-domain of artificial neural networks, which utilizes the benefits of content-addressable memory (CAM) in microcomputers. The AM concept was originally developed in an effort to utilize the power and speed of existing computer systems for solving large-scale and computationally intensive problems by simulating biological neurosystems.

Hierarchical Graph Neuron (HGN) approach [20] is a type of associative memory, which signifies the hierarchical structure in its implementation. Hierarchical structure in associative memory models are of interest as this has been shown to improve the rate of the recall in pattern recognition applications. The distributed HGN scheme also allows for better control of the network resources. This scheme compares well with contemporary approaches such as Self-Organizing Map and Support Vector Machine in terms of speed and accuracy. Vector Symbolic Architecture [6, 21] is a class of connectionist models that use hyper-dimensional vectors (i.e. vectors of several thousand elements) to encode structured information as distributed or holographic representation. In this technique

structured data is represented by performing basic arithmetic operations on field-value tuples. Distributed representations of data structures is an approach actively used in the area of cognitive computing for representing and reasoning upon semantically bound information [21]. VSA-based knowledge-representation architecture was also proposed for learning arbitrarily complex, hierarchical, symbolic relationships (patterns) between sensors and actuators in robotics. Recently the theory of hyper-dimensional computing was adopted for implementing communication protocols for collective communications in machine-to-machine communication scenarios [22].

Shortcomings of the traditional data driven machine-learning based fault identification

The traditional machine learning based data driven fault detection systems includes the following phases: Data collection; Training; and Inference, i.e. identification of faults based on the presented fresh data. During the data collection phase statistics of signal measurements from different system's components is gathered. The statistics could either be obtained from historical logs or from system-level simulations. In the supervised training phase the set of measurements corresponding to a particular fault is presented to the particular machine-learning algorithm. In many practical cases with large complex systems-of-systems the interdependencies between different measured signals are not known or require complex analysis, the entire set of signals is, therefore, used for the training, as illustrated in Fig. 1. However, for traditional machine-learning algorithms there are two main problems associated with this choice.

Firstly, most of the traditional machine-learning methods have different limitations on the number of features, which could be used for the robust classification. The model-based classifiers (e.g. logical regression) are known for their degraded convergence and computational performance when the number of features is growing. In the knowledge-based and specifically the direct pattern matching based classification approaches (e.g. Artificial Neural Networks) the size of the training data set grows dramatically with the increase in the number of features.

The second problem comes with the inherent centralized nature of the data driven approaches. Indeed, in order to robustly classify a fresh input pattern of signals the system needs to be presented with a full pattern. Presenting only a part of a pattern would substantially decrease the classification accuracy unless the partial pattern was present in the training data set. In Section 6 we discuss how this problem can be overcome.

3 The outline of the approach

3.1 Bio-inspired pattern matching based fault identification

The approach presented in this article also uses approach to data collection presented in Fig. 1. In this model each cell in the grid denotes a state of the particular component of the plant and/or of the automation system (in the article each cell is also referred

to as a *node*). It is supposed that each component can inform a central unit about its current state. A *state* is the quantitative characterization of a certain parameter of the component computed over a particular time interval. In the scope of simulations in this paper state is understood as the statistics (*MIN*, *MAX*, *MEAN*) over the historical values of the particular parameter computed over a predefined time window.

In the proposed approach the inference of the potential fault is done by the central unit, which collects states from all N components. For this patterns of the system-wide state corresponding to different faults are stored in the unit. In order to construct the pattern the unit anchors components states at the moment of the pattern construction.

The core of the proposed approach is Holographic Graph Neuron (HoloGN) [23, 24] – a one-shot learning associative memory for pattern recognition. The principles of HoloGN are exemplified in Fig. 1.

Graph Neurons (GNs), shown as a matrix in the middle part of Fig. 1 is an abstraction model for memorizing patterns of generic sensor stimuli for later template matching. Depending on the application domain GNs could be interpreted differently. For example, when GN is used for analyzing character patterns in strings of fixed length, a particular GN is responsible to responding to symbols appearing on a specific pre-assigned position in the string. In the case of the proposed approach each GN encodes the states of the particular system component.

Hierarchical Graph Neuron [20] aggregates the holistic view of the entire pattern through a hierarchy of graph neurons. In contrast to the contemporary machine learning approaches, HGN allows induction of new patterns in the learning set without the need for re-training. Whilst doing so it exhibits a high level of scalability i.e. its performance and accuracy do not degrade as the number of stored pattern increases over time.

HoloGN, shown as a colored equation and the binary code in the upper part of Fig. 1, is an approach for encoding the activated neurons and their interrelation within a pattern using special type for data representation called Vector Symbolic Architectures (VSA) [21]. Using VSA for encoding the hierarchy of graph neurons allows for the streamlined implementation of the associative memory and efficient processing while maintaining the high level of inference accuracy.

The proposed approach functions in two phases: the learning phase and the fault identification phase. The learning phase could either be performed off-line and using system-level simulation facilities or in real time learning the fault situation through an interaction with a SCADA system and a human operator. This article focuses on the off-line learning phase in order to demonstrate the suitability of the proposed approach and its unique performance properties. The result of the learning phase is a collection of patterns of system-wide state corresponding to the particular fault encoded using HoloGN approach.

In the fault identification first each component informs the central unit about its current state. The pattern of states is then encoded using HoloGN. Finally the recall operation is performed on the collection of patterns from learning phase. The result of the recall is a list of potential faults.

4 Details of the proposed approach

4.1 Theoretical preliminaries

Vector Symbolic Architecture is an approach for encoding and operations on distributed representation of information. VSA is so far mainly used in the area of cognitive computing for representing and reasoning upon semantically bound information [6, 21].

The fundamental difference between distributed and localist representations of data is as follows. In traditional (localist) computing paradigm each bit and its position within a structure of bits are significant. For example a field in a database has a predefined offset amongst other fields and a symbolic value has unique representation in ASCII codes. In the distributed representation all entities are represented by binary random vectors of very high dimension also called Binary Spatter codes. Further in the article for brevity reasons HD-vector term is used when referring to hyperdimension vectors. Hyperdimensionality means here several thousand of binary positions for representing a single entity. In [21] it is proposed to use vectors of 10000 binary elements.

Similarity Metric

A similarity between two binary representation is characterized by Hamming distance, which measures the number of positions in two compared vectors in which they are different: $\Delta_H(A, B) = \frac{1}{d}\|A \otimes B\|_1 = \frac{1}{d}\sum_{i=0}^{d-1} a_i \otimes b_i$, where a_i, b_i are bits on positions i in vectors A and B of dimension d, and where $\otimes$ denotes the bit-wise XOR operation.

Randomness

Randomness means that the values on each position of an HD-vector are independent of each other, and "0" and "1" components are equally probable $p_1 = p_0 = 0.5$. The statistical properties of these HD-vectors are described by the binomial distribution: $\Pr(k, d, p) = \binom{d}{k} p^k (1-p)^{d-k}$. According to the last property, on very high dimensions d, the distances from any arbitrary chosen HD-vector to more than 99.99% of all other vectors in the representational space are concentrated around 0.5 Hamming distance.

Generation of HD-vectors

Random binary vectors with the above properties could be generated based on Zadoff-Chu sequences [25]. Using this principle a sequence of K pseudo-orthogonal vectors to a given initial random HD-vector $\mathbf{A}$ is obtained by cyclically shifting $\mathbf{A}$ on i positions, where $1 < i < d$. Further in the article this operation is denoted as $\mathrm{Sh}(\mathbf{A}, i)$.

Bundling of vectors

Joining several entities into a structure is done by a bundling operation. It is implemented by a thresholded sum of the HD-vectors representing entities. A bit-wise thresholded sum of n vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. Further

term "majority sum" is used and denoted as $[\mathbf{A} + \mathbf{B} + \mathbf{C}]$. The relevant properties of the majority sum are:

- The result is a random vector, i.e. the number of "1" components is approximately equal to the number of "0" components;
- The result is similar to all vectors included in the sum;
- Number of vectors in majority sum must be odd;
- The more HD-vectors involved into majority sum operations the closer is Hamming distance between the resultant vector and any HD-vector component to 0.5;
- If several copies of any vector are included into majority sum the resultant vector is closer to the dominating vector than to other components.

The algebra on VSA includes other operations e.g., binding and permutation. Since in the scope of this article they are not used the description of their properties is omitted.

4.2 Holographic Graph Neuron

In this section we overview principles for forming of Holographic Graph Neuron using Vector Symbolic Architectures. In the case of HoloGN, all its elements (i.e. symbols of individual neurons' alphabet) are indexed uniquely and the index of a particular element is derived as a function of the GN's ID. Let IV_j is an initialization HD-vector for GN j. The initialization vectors for different GNs are chosen to be mutually orthogonal. Then the HD-index of element i in GN j is computed as $E_{j,i}^{HD} = Sh(IV_j, i)$.

Let N be the number of individual Graph Neurons. When a GN array is exposed to a pattern of length N, the holographic representation of the activated elements is formed as: $HGN = [\sum_{j=1}^{N} E_j^{HD}]$, where E_j^{HD} is the HD-index of the activated element in GN j. The majority sum operation is then applied as described above.

4.3 Adoption of HoloGN for encoding system state

In the proposed method HoloGN architecture is applied to encode the components states at the central unit. Referring to Fig. 1, each component is assigned a system-wide unique initialization HD-vector IV.

The aggregated system state is constructed using the historical data by encoding all states leading to the particular fault into a single HoloGN representation: $HGN_{FAULT1} = [\sum_{j=1}^{N} STATE_j^{HD}]$, where N in this case is the number of system's components and $STATE_j^{HD}$ is the HoloGN-encoded state of component j, which contributes to a system-wide fault FAULT1. In the case of several historical cases leading to the particular fault i, the corresponding aggregated system states for this fault are bundled together as: $HGN_{FAULT^i} = [HGN_{FAULT^i}^1 + HGN_{FAULT^i}^2 + \cdots + HGN_{FAULT^i}^M]$, where M is the number of encountered cases leading to fault i.

4.4 Fault identification in the proposed approach

The fault identification procedure has its foundation in the statistical properties of the hyperdimensional random binary vectors used for encoding states of system's components and the similarity property of the majority sum is used for bundling of several vectors together.

The joint system state HGN_{FAULT^i} is an associative memory of all combinations of system-wide states, which characterize the particular fault. The fault identification is, therefore, performed by testing the inclusion of the current pattern of system states with holographic representations for different faults HGN_{FAULT^i}. This is done by computing the Hamming distance $\Delta_H(HGN_{Current}, HGN_{FAULT^i})$, between the HoloGN encoded current system state $HGN_{Current}$ and the states of all possible faults HGN_{FAULT^i}. The closer this metric to 0.5 for the particular fault the less likely is that the current state is an indication of this fault.

5 Performance benchmarking with related approaches

The case study used for demonstrating the proposed cognitive fault detection system is a generic nuclear power plant model provided by Fortum Power and Heat, a power utility with nuclear power plant operation license in Finland. The Apros 6 process simulator is used to run the model. Apros 6 is a dynamic process simulator owned by the VTT Technical Research Centre of Finland and Fortum. Two main process loops are used for power generation using nuclear energy, the primary and secondary circuit. The primary circuit contains the reactor vessel and the nuclear fission within the fuel generates thermal energy, which heats the water in the vessel. The coolant pumps in the primary circuit circulate water through the steam generators and the reactor vessel and thus thermal energy is transferred from the primary to the secondary circuit. The pressurizer, also part of the primary circuit, is a vessel is partially filled with water and it is designed for pressure regulation using heaters and water sprays. The secondary circuit is connected to the primary through the steam generators. Heat from the primary circuit converts water flowing in the secondary side of the steam generators into steam. Turbines use the high-pressure steam flow and drive electric generators. Condensers are used to convert the low-pressure steam after the turbines back to water.

5.1 Data sources

The functional failure identification and propagation framework was used to analyze 116 automation components as the sources of hardware faults [26]. Most of these components were pump and valve actuators. For each automation component type three specific failure modes were chosen (e.g. a valve actuator can be set to the "failed open", "failed closed" or "no electric supply" fault mode which will result in the opening, closing or stop controlling the valve). A component – failure mode pair defines a fault in this paper (e.g., "Valve valveID" – "Fails - Open"). From the 348 total possible faults (116 components x

3 failure modes per component), 92 faults actually affected the steady state operation of the power plant model and thus can be detected by a data driven fault detection system.

In the case study the model was driven to 11 power production levels in order to get more simulation data for the set of faults. The 92 faults, which are detectable in the steady state of the plant, are simulated for every one of the 11 power levels (1012 simulations in total). The results of these simulations were used to build data sets for training and testing the fault detection systems. Each simulation (fault - power level) contains 180 seconds of simulation time. Data set entries are generated by 37 monitored simulation signals. Three statistical values are generated per signal leading to a 111 inputs per entry (37 signals with 3 statistical values per signal). The data sets had a total of 1012 entries, each with 111 inputs (generated by the 37 logged signals) and at the end every entry contains the classification attribute.

5.2 Reference technologies and benchmarking tools

The performance of the proposed approach was compared to the performance of a multilayer perceptron Artificial Neural Network (ANN), a decision tree, and a K-nearest-neighbors (KNN) classifier. The WEKA tool and MATLAB were used to benchmark the accuracy of the fault detection of the proposed method. Note that the decision tree and ANN produce a single prediction for a given input. The KNN classifier and the proposed approach produce a ranked list of possible results. During the benchmarking process an accuracy of the fault identification was assessed. The accuracy was measured as the ratio of the correctly identified faults over all presented cases from the test set.

5.3 Calibrating the benchmarking technologies

Since not all methods handle normalized data equally well, the all values in the training and the test data sets were normalized. The data values were mapped to the range [-1,1].

The accuracy of classification by the proposed approach depends on the quantization level when feeding the state values to the input of the technologies under benchmarking. Cross-validation suggests that the optimal value is sixty levels The KNN method produced best prediction results for the number of the nearest neighbors equals 3. The ANN was feedforward with two inputs and output layers no hidden layers and back propagation with momentum algorithm was used for training. The J48 algorithm, which is based on the C4.5 algorithm, was used to train the decision tree.

5.4 Results of comparison

To the best of the authors' knowledge benchmarking machine learning techniques operate only in the centralized mode. That is the data set presented for the fault detection should have the same structure as the training data set. In this article a zero knowledge about the underlying component interdependencies is assumed. Therefore all the compared approaches were trained by presenting the state values from all system components as in Fig. 1.

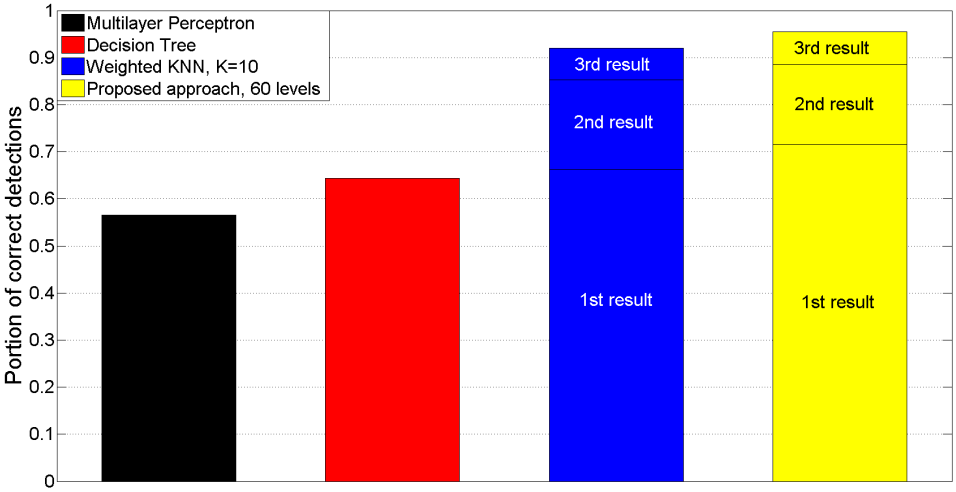


Figure 2: The results of benchmarking: ANN, decision tree, KNN and the proposed approach.

The results of comparison of the considered approaches are shown in Fig. 2. The main result indicated by the figure is that the accuracy of the fault identification by the proposed approach outperforms the other approaches even when considering the top one result. Namely the best ANN's accuracy index was nearly 0.6; the accuracy of the decision tree was 0.65. The best accuracy was demonstrated by both KNN classifier and the proposed approach (0.68 and 0.71 correspondingly). When considering top three results both KNN and our method show the accuracy above 0.9 with the accuracy of the proposed approach achieving 0.95.

The above presented result is a very positive one for the approach. Showing either matching or in most of the cases superior performance over the traditional approaches our approach has one more unique advantage: it is suitable for the distributed implementation. However, the proposed approach does not necessarily always outperform the benchmarked techniques, e.g. ANN is very powerful for classification when borders between classes are non-linear.

6 Discussion and Conclusions

This article presented an intelligent, biologically inspired fault detection method. The proposed methodology utilizes the concepts of associative memory and Vector Symbolic Architectures, commonly used for modeling cognitive abilities of human brain. The proposed approach features excellent fault detection accuracy compared to that of the traditional machine learning methods used in the same context.

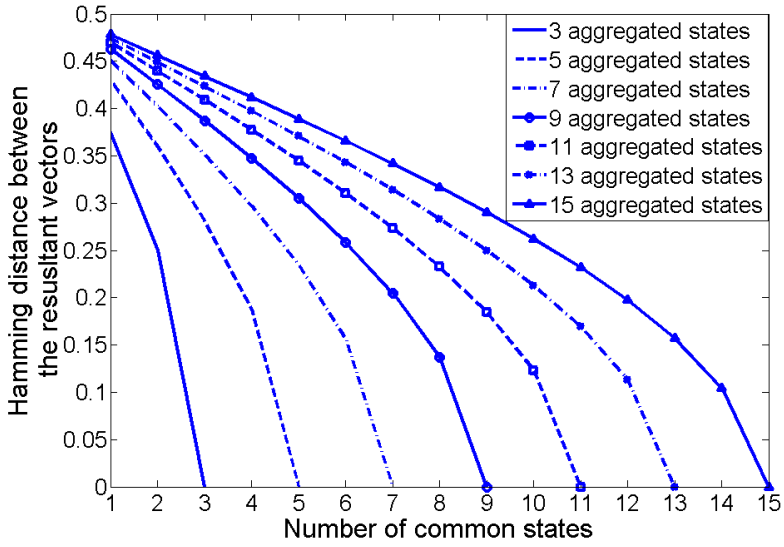


Figure 3: Hamming distance as an indicator of presence of partial state in the joint representation of a system-wide state.

At the same time the results presented in Fig. 3 [24] promise that the proposed approach could be suitable for the distributed implementation. Fig. 3 shows that Hamming distance can serve as a robust indicator of the presence of distributed representation of the vectors including particular partial states of the system-wide states characterizing the particular fault. Thus the detection could be done without the centralized collection of the system-wide states in the operation mode. This would make the implementation of the fault diagnostic subsystem an integral part of the distributed automation system of the plant, enabling its deployment in the standard programmable logic controllers. The future development of the proposed approach includes its extension to a fully distributed version, implementation of the approach with IEC 61499 and multi-agent systems and performance benchmarking with the centralized approach presented here. The second direction for future work is detection of combinatorial faults. The proposed approach promises a high performance compare to the traditional ones. Distributed design along with performance benchmarking for combinatorial faults will be reported elsewhere.

Acknowledgements

This work is partially supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), institutional grant IG2011-2025, the grant 381940 of Luleå University of Technology, and SAUNA project of the The Finnish Research Programme on Nuclear Power Plant Safety 2015 – 2018 (SAFIR2018) program.

References

[1] I. E. Commission, "International Standard IEC61499-1," in *Function Blocks - Part 1 Architecture, ed. Geneva*, 2005.

[2] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review," *Industrial Informatics, IEEE Transactions on*, vol. 7, pp. 768 – 781, 2011.

[3] J. Yan and V. Vyatkin, "Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 2200–2209, 2013.

[4] G. Zhabelova, V. Vyatkin, and V. Dubinin, "Towards Industrially Usable Agent Technology for Smart Grid automation," *Industrial Electronics, IEEE Transactions on*, vol. 62, no. 4, pp. 2629–2641, 2015.

[5] A. Mousavi and V. Vyatkin, "Energy Efficient Agent Function Block: a semantic agent approach to IEC 61499 function blocks in energy efficient building automation systems," *Automation in Construction, Elsevier*, vol. 54, pp. 127–142, 2015.

[6] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Computation*, vol. 25, no. 8, pp. 2038–2078, 2013.

[7] F. J. Alexander, "Machine Learning [Guest Editor's introduction]," *Computing in Science and Engineering*, vol. 15, pp. 9–11, 2013.

[8] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers and Chemical Engineering*, vol. 27, pp. 293–311, 2003.

[9] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies," *Computers and Chemical Engineering*, vol. 27, pp. 313–326, 2003.

[10] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part III: Process history based methods," *Computers and Chemical Engineering*, vol. 27, pp. 327–346, 2003.

[11] R. Isermann, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control*, vol. 29, pp. 71–85, 2005.

[12] V. Uraikul, C. W. Chan, and P. Tontiwachwuthikul, "Artificial intelligence for monitoring and supervisory control of process systems," *Engineering Applications of Artificial Intelligence*, vol. 20, pp. 115–131, 2007.

[13] Z. Kurd, T. Kelly, and J. Austin, "Developing artificial neural networks for safety critical systems," *Neural Computing and Applications*, vol. 16, pp. 11–19, 2007.

[14] S. J. Lee and P. H. Seong, "A dynamic neural network based accident diagnosis advisory system for nuclear power plants," *Progress in Nuclear Energy*, vol. 46, pp. 268–281, 2005.

[15] M. J. Embrechts and S. Benedek, "Hybrid identification of nuclear power plant transients with artificial neural networks," *Industrial Electronics, IEEE Transactions on*, vol. 51, pp. 686–693, 2004.

[16] T. Amraee and S. Ranjbar, "Transient Instability Prediction Using Decision Tree Technique," *Power Systems, IEEE Transactions on*, vol. 28, pp. 3028–3037, 2013.

[17] D. Xuewu and G. Zhiwei, "From Model, Signal to Knowledge: A Data-Driven Perspective of Fault Detection and Diagnosis," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 2226–2238, 2013.

[18] J. Neuzil, O. Kreibich, and R. Smid, "A Distributed Fault Detection System Based on IWSN for Machine Condition Monitoring," *Industrial Informatics, IEEE Transactions on*, vol. 10, pp. 1118–1123, 2014.

[19] Z. Yingwei, Z. Hong, S. J. Qin, and C. Tianyou, "Decentralized Fault Diagnosis of Large-Scale Processes Using Multiblock Kernel Partial Least Squares," *Industrial Informatics, IEEE Transactions on*, vol. 6, pp. 3–10, 2010.

[20] B. B. Nasution and A. I. Khan, "A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition," *Neural Networks, IEEE Transactions on*, vol. 19, no. 2, pp. 212–229, February 2008.

[21] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation, Springer*, vol. 1, no. 2, pp. 139–159, October 2009.

[22] D. Kleyko, N. Lyamin, E. Osipov, and L. Riliskis, "Dependable mac layer architecture based on holographic data representation using hyper-dimensional binary spatter codes," in *Multiple Access Communications : 5th International Workshop, MACOM 2012*. Springer, 2012, pp. 134–145.

[23] E. Osipov, A. I. Khan, and A. Anang, "Holographic graph neuron," in *Computer and Information Sciences (ICCOINS), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[24] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Sekercioglu, "Holographic Graph Neuron: a Bio-Inspired Architecture for Pattern Processing," *Preprint at http://arxiv.org/pdf/1401.0742v1.pdf*, pp. 1–9, 2015.

[25] D. Kleyko and E. Osipov, "On bidirectional transitions between localist and distributed representations: The case of common substrings search using vector symbolic architecture," *Procedia Computer Science*, vol. 41, pp. 104–113, 2014.

[26] N. Papakonstantinou, S. Proper, B. O'Halloran, and I. Y. Tumer, "Simulation Based Machine Learning For Fault Detection In Complex Systems Using The Functional Failure Identification And Propagation Framework," in *ASME CIE, Buffalo NY, USA*, 2014, pp. 1–10.